MODELING AND SOLVING MULTI-PRODUCT MULTI-LAYER LOCATION-ROUTING PROBLEMS

A Dissertation
Submitted to the Graduate Faculty
of the
North Dakota State University
of Agriculture and Applied Science

By

Mohsen Hamidi

In Partial Fulfillment of the Requirements
for the Degree of
DOCTOR OF PHILOSOPHY

Major Department:
Industrial and Manufacturing Engineering

November 2011

Fargo, North Dakota

North Dakota State University
Graduate School

Title

Modeling and Solving Multi-Product Multi-Layer Location-Routing Problems

By

Mohsen Hamidi

The Supervisory Committee certifies that this ***disquisition*** complies with North Dakota State University's regulations and meets the accepted standards for the degree of

DOCTOR OF PHILOSOPHY

ABSTRACT

Hamidi, Mohsen, PhD, Department of Industrial and Manufacturing Engineering, College of Engineering and Architecture, North Dakota State University, November 2011. Modeling and Solving Multi-Product Multi-Layer Location-Routing Problems. Major Professor: Dr. Kambiz Farahmand.

Distribution is a very important component of logistics and supply chain management. Location-Routing Problem (LRP) simultaneously takes into consideration location, allocation, and vehicle routing decisions to design an optimal distribution network. Multi-layer and multi-product LRP is even more complex as it deals with the decisions at multiple layers of a distribution network where multiple products are transported within and between layers of the network. This dissertation focuses on modeling and solving complicated four-layer and multi-product LRPs which have not been tackled yet. The four-layer LRP represents a multi-product distribution network consisting of plants, central depots, regional depots, and customers. The LRP integrates location, allocation, vehicle routing, and transshipment problems.

Through the modeling phase, the structure, assumptions, and limitations of the distribution network are defined and the mathematical optimization programming model that can be used to obtain optimal solutions is developed. Since the mathematical model can obtain the optimal solution only for small-size problems, through the solving phase metaheuristic algorithms are developed to solve large-size problems. GRASP (Greedy Randomized Adaptive Search Procedure), probabilistic tabu search, local search techniques, the Clarke-Wright Savings algorithm, and a node ejection chains algorithm are combined to solve two versions of the four-layer LRP. Results show that the metaheuristic can solve the problem effectively in terms of computational time and solution quality. The

presented four-layer LRP, which considers realistic assumptions and limitations such as producing multiple products, limited plant production capacity, limited depot and vehicle capacity, and limited traveling distances, enables companies to mimic the real world limitations and obtain realistic results. The main objective of this research is to develop solution algorithms that can solve large-size multi-product multi-layer LRPs and produce high-quality solutions in a reasonable amount of time.

ACKNOWLEDGEMENTS

First and foremost I should thank my advisor, Professor Kambiz Farahmand, for his support, encouragement, and advice during my PhD study. Professor Farahmand has had a great role in my success and achievements in teaching and research.

I would like to express my thanks to Dr. Reza Sajjadi for his great help from the beginning of my research. I could not perform this research without his help.

I should acknowledge Professor Kendall Nygard's support and advice during my research. I also thank Dr. Ababei, Dr. Zhang, and Dr. Maleki for their comments and suggestions.

DEDICATION

This research is dedicated to my parents and my family.

My parents have a great share in my success and achievements.

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

LIST OF TABLES

LIST OF FIGURES

LIST OF FIGURES (Continued)

LIST OF FIGURES (Continued)

CHAPTER 1. INTRODUCTION

Distribution is a very important component of logistics and supply chain management. Distribution in supply chain management refers to the distribution of products from producer to wholesalers, retailers, or consumers through distribution hubs, e.g. depots and warehouses. This process is known as the distribution network or chain. Decreasing the distribution cost leads to decreasing the final cost of the product, so companies try hard to lower their distribution costs to remain strong in the competitive market and compete with their competitors. Designing new efficient distribution networks and improving existing distribution networks are keys to distribution cost reduction.

The Location-Routing Problem

The Location-Routing Problem (LRP) is a relatively new branch of location analysis that takes into account vehicle routing aspects (Nagy and Salhi, 2007). LRP simultaneously takes into consideration location, allocation, and vehicle routing decisions to design a distribution network. These decisions are the key decisions in distribution network design. The location problem involves selecting locations of facilities, through which products are transported to customers or other facilities, from a set of candidate sites. The allocation problem deals with assigning customers to the selected facilities. The Vehicle Routing Problem (VRP) is a problem in which a set of routes for a fleet of vehicles based at one or several depots must be determined for a number of customers (The VRP Web, 2010). The objective of the VRP is to deliver a set of customers with known demands on minimum-cost vehicle routes originating and terminating at a depot (The VRP Web,

2010). The LRP solves the joint problem of determining the optimal number, capacity, and location of facilities serving more than one customer and finding the optimal set of vehicle schedules and routes (Min et al., 1998). The common objective for LRPs is to minimize the overall cost that includes depot costs and transportation costs. Its major aim is to capitalize on distribution efficiency resulting from a series of coordinated, non-fragmented movements and transfer of goods (Min et al., 1998). The main difference between the LRP and the classical location-allocation problem is that once the facility is located, the former requires customer visitation through tours, whereas the latter assumes the straight-line or radial trip from the facility to the customer (Min et al., 1998). LRPs are often described as a combination of three distinct components: (i) facility location, (ii) allocation of customers to facilities, and (iii) vehicle routing (Laporte, 1988). These three sub-problems are closely interrelated and cannot be optimized separately without running the risk of arriving at a suboptimal solution (Laporte, 1988).

From a practical viewpoint, location-routing forms part of distribution management (Nagy and Salhi, 2007). The LRP models can be applied in a variety of businesses and industries. Some of the applications mentioned in Nagy and Salhi (2007) are food and drink distribution, newspaper distribution, parcel delivery, and waste collection. Although most of the LRP models focus on distribution of consumer goods or parcels, there are also some applications in healthcare (e.g. blood bank location), the military (e.g. military equipment location), and communications (e.g. telecommunication network design) (Nagy and Salhi, 2007). Operational research is all too often applied only in the affluent countries of Western Europe and North America, thus it is pleasing to see that LRP has also been applied in developing countries (Nagy and Salhi, 2007). An example of an LRP model is

shown in Figure 1.1. The distribution system is a three-layer LRP: a plant, warehouses, and stores. The product is produced in the plant, shipped to warehouses, and then delivered to stores. In some LRPs, e.g. this distribution system, customers are wholesalers or retailers and in some, e.g. a mail delivery system, customers are the final customers. To design the distribution network, the main decisions to be made are as follows:

- Given a set of candidate sites, how many warehouses are needed and where should they be located?
- Which customers (stores) should be allocated to which warehouse?
- How many tours should exist for each warehouse, which customers should be on each tour, and what is the best sequence of the customers on each tour?
- How many products should be shipped from the plant to each warehouse?

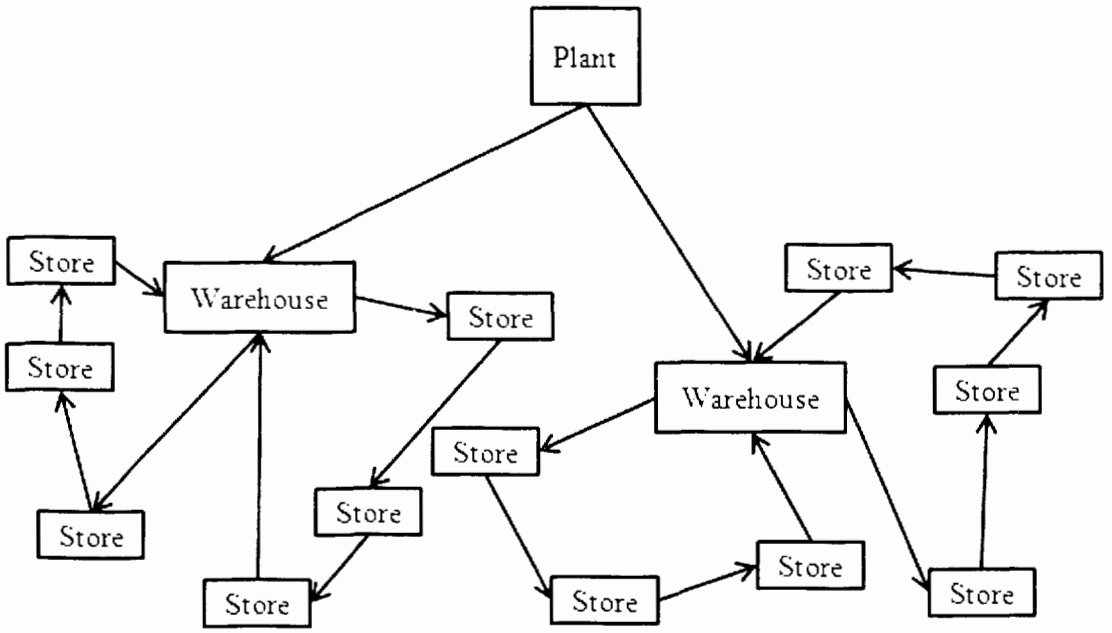

Figure 1.1. A distribution system.

Solution Methods

From a mathematical point of view, the LRP is a combinatorial optimization problem (Nagy & Salhi, 2007). In combinatorial optimization problems, the number of solutions grows exponentially with the size of problem and evaluating all of the solutions is not possible in a reasonable amount of time (Feo & Resende, 1995). LRP mathematical models are usually mixed integer linear programming models or mixed integer nonlinear programming models. To solve different versions of the LRP, two types of solution methods have been developed; exact methods and heuristic methods. Exact algorithms for the LRP can be classified into four categories: (1) direct tree search/branch-and-bound, (2) dynamic programming, (3) integer programming, and (4) non-linear programming (Min et al., 1998).

Comprehensive mathematical programming formulations which incorporate simultaneously all aspects of the LRP problem will, in general, contain too many variables and constraints to be easily solved using an exact algorithm (Laporte, 1988). The LRP is an NP-hard (Non-deterministic Polynomial-time hard) problem, as it encompasses two NP-hard problems (facility location and vehicle routing) (Nagy and Salhi, 2007). The computational effort required to solve NP-hard problems increases exponentially with the problem size (The VRP Web, 2010). For such problems it is often desirable to obtain approximate solutions, so they can be found fast enough and are sufficiently accurate for the purpose (The VRP Web, 2010). Usually this task is accomplished by using various heuristic methods, which rely on some insight into the problem nature (The VRP Web, 2010). An example is the model developed by Perl and Daskin (1985). Their model is a three-layer LRP consisting of supply sources, warehouses, and customers. For a problem

with 1 supply source, 3 potential warehouse locations, and 14 customers, the formulation includes 1201 integer variables and 8455 constraints (Perl and Daskin, 1985). Increasing the number of customers to 100 would increase the number of integer variables to 265,528 and the number of constraints to over $6.33*10^{29}$ (Perl and Daskin, 1985). In fact, due to the complexity of LRP, exact methods can only tackle relatively small instances (problems with small number of facilities and customers) (Nagy and Salhi, 2007). For solving larger problems and real instances, the only helpful methods have been heuristics (Ambrosino and Scutella, 2005). The most common techniques used in LRP heuristics are simulated annealing, tabu search, greedy randomized adaptive search procedure (GRASP), genetic algorithm, neural networks, and ant colony.

CHAPTER 2. LITERATURE REVIEW

Balakrishnan et al. (1987), Laporte (1988), Min et al. (1998), and Nagy and Salhi (2007) have surveyed the LRP models. In this literature review, the LRP models are categorized into two-layer LRPs and multi-layer (three- and four-layer) LRPs.

Two-Layer LRPs

Most of the LRP models proposed in the literature are related to a distribution network with two layers of depots and customers (Ambrosino and Scutella, 2005). In these models, three problems are to be solved: 1) location problem: given a set of candidate sites, how many depots are needed and where should they be located?, 2) allocation problem: which customers should be allocated to which depot?, and 3) routing problem: how many tours should exist for each depot, which customers should be on each tour, and what is the best sequence of the customers on each tour?. Many studies have been performed on two-layer LRPs of which some are reviewed in this section.

Perl (1983) and Perl and Daskin (1985) formulated a three-layer LRP (with plants, warehouses, and customers) but presented a heuristic method to solve a two-layer LRP with capacitated depots and vehicles. The heuristic solution method is based on decomposing the LRP into three sub-problems and solving the sub-problems in a sequential and iterative manner while accounting for the dependence between them. The sub-problems are: (1) Mutli-Depot Vehicle-Dispatch Problem (MDVDP), (2) Warehouse Location-Allocation Problem (WLAP), and (3) Multi-Depot Routing-Allocation Problem

(MDRAP). MDVDP assumes that all potential warehouse sites are used and constructs an initial set of routes. This set of routes (suppressing the linkages to warehouses) is used as input to WLAP. WLAP locates the warehouses and allocates the routes constructed in MDVDP or MDRAP to them. The selected warehouse locations are used as input to MDRAP. MDRAP simultaneously reallocates the customers to warehouses and designs delivery routes for the warehouses selected in WLAP. The output from MDRAP provides a new set of delivery routes which are used as input to a new iteration through WLAP. Heuristic methods were developed to solve MDVDP and MDRAP, and an exact method was applied to solve WLAP. The heuristic methods consist of a basic algorithm and three improvement procedures. The basic algorithm uses a sequential savings method.

Hansen et al. (1994) modified the method developed by Perl (1983) and Perl and Daskin (1985) and presented a modified heuristic method to solve a two-layer LRP with capacitated depots and vehicles. They solved all of the three sub-problems (MDVDP, WLAP, and MDRAP) introduced by Perl (1983) and Perl and Daskin (1985) heuristically in an iterative manner. The heuristic to solve MDVDP generates an initial solution and then improves the solution by applying 2-optimal, exchange, and single displacement algorithms. The heuristic to solve WLAP allocates all routes to one depot without considering the depot capacity and then tries to move the routes to other depots by applying route cluster displacement, single route displacement, forced depot closing, and depot opening sequence algorithms. The heuristic to solve MDRAP generates an initial solution and then improves the solution by applying customer orientated initiatives, extended node exchange, and two by two node displacement algorithms. They also compared their

solutions with Perl's (1983) solutions and showed that their heuristic method produces better solutions.

Tuzun and Burke (1999) presented a two-phase tabu search algorithm that iterates between location and routing phases in order to search for better solutions of a two-layer LRP with capacitated vehicles. The approach coordinates two tabu search mechanisms: one seeking a good facility configuration and the other a good routing that corresponds to this configuration. They compared the algorithm with SAV1 heuristic introduced by Srivastava (1993) and found out that their algorithm is better than SAV1 algorithm (2.61% savings over SAV1 algorithm).

Wu et al. (2002) presented a decomposition-based method for solving a two-layer LRP with multiple fleet types and a limited number of vehicles for each different vehicle type. The problem is decomposed into two sub-problems (the location-allocation problem, and the vehicle routing problem) and then each sub-problem is solved in a sequential and iterative manner by a combined simulated annealing and tabu search framework. They used three problems from Perl's (1983) study and compared the results of Perl's (1983) method, Hansen et al.'s (1994) method, and their proposed method to test the performance of their method. Their heuristic produced an optimal solution for the test problem with 12 nodes (just like Perl's (1983) and Hansen et al.'s (1994) studies). For the problem with 55 nodes, it outperformed both Perl's (1983) and Hansen et al.'s (1994) methods. For the problem with 85 nodes, the proposed method provided a better solution than Perl's (1983) but Hansen et al.'s (1994) method produced a better solution than the proposed method. The proposed method was also tested over a number of newly generated problems.

Prins et al. (2006b) presented a heuristic to solve a two-layer LRP with capacitated depots and vehicles. The method is a greedy randomized adaptive search procedure (GRASP) with learning process based on a randomized and extended version of Clarke and Wright algorithm, followed by a path relinking procedure as post-optimization. They compared their method (GRASP) with three heuristics on three sets of instances. They showed that on average, GRASP saves 3.77% in comparison with a multi-start local search heuristic proposed by Prins et al. (2004). They compared GRASP with a cluster based heuristic (CL) and a lower bound proposed by Barreto (2004) and found out that GRASP is not farther than 0.28% of the optimal solution on average when this one is known, while CL is at 1.2%. Also, in most cases, GRASP's solutions are closer to the lower bound than CL. They also showed that GRASP is competitive with the two-phase tabu search (TS) proposed by Tuzun and Burke (1999) since its average deviation to the TS algorithm is -0.77% and it improves 26 solutions out of 36 (72%).

Barreto et al. (2007) introduced a cluster analysis based sequential heuristic for solving a two-layer LRP with capacitated distribution centers and vehicles. The heuristic has four steps: 1) constructing groups of customers with a capacity limit, 2) determining the distribution route in each customer group, 3) improving the routes, and 4) locating the distribution centers and assigning the routes to them. Moreover, four clustering techniques and six proximity measures were used to obtain several versions of the heuristic. The computational tests indicate that the one-phase hierarchical clustering technique has a slightly better performance than the other versions. Concerning proximity measures, the group average measure has produced the most balanced results. They also compared the results of the heuristic with lower bounds obtained from a relaxed 2-index integer linear

programming formulation presented by Barreto (2004). For all the instances, the gap falls between a minimum of 0% and a maximum of 19.01% with an average of 4.81% and a median of 3.13%.

Prins et al. (2007) presented a Lagrangean Relaxation-Granular Tabu Search (LRGTS) heuristic to solve a two-layer LRP with capacitated depots and vehicles. The LRGTS performs one location phase and one routing phase in each iteration. In the first phase, the routes and their customers are aggregated into super-customers, leading to a facility-location problem, which is then solved by a lagrangean relaxation of the assignment constraints. In the second phase, the routes from the resulting multi-depot vehicle-routing problem are improved using a granular tabu search heuristic. The algorithm also tries to further improve the solution by performing a local search phase. At the end of each iteration, information about the edges most often used is recorded to be used in the following phases. The LRGTS is evaluated on three sets of instances and compared with GRASP (Prins et al., 2005), clustering heuristic (CH) (Barreto, 2004 or Barreto et al., 2007), and TS (Tuzun and Burke, 1999). The results show that LRGTS performs better than the other three heuristics.

Marinakis and Marinaki (2008a) presented a bi-level formulation for a two-layer LRP with capacitated depots and vehicles. The formulation separates the problem into two problems: the capacitated facility location problem and the vehicle routing problem. To solve the problem, the authors developed a bi-level metaheuristic algorithm. Genetic algorithm, Greedy Randomized Adaptive Search Procedure (GRASP), and Neighborhood Search techniques are integrated in the metaheuristic algorithm. In the first level of the algorithm, the capacitated facility location problem is solved to locate facilities and assign

customers to the facilities. In the second level, given the solution of the first level, the vehicle routing problem is solved to obtain the routing of the vehicles. The algorithm was tested on a set of benchmark problems. The results were satisfactory as the algorithm found new best solutions for some of the benchmark problems.

Duhamel et al. (2010) proposed the GRASP×ELS heuristic for solving a two-layer LRP with capacitated depots and vehicles. The heuristic is a greedy randomized adaptive search procedure (GRASP) hybridized with an evolutionary local search (ELS). The method builds giant tours and then splits them into feasible routes by using a splitting procedure. They compared the GRASP×ELS with three heuristics (GRASP (Prins et al., 2006b), MA|PM (Prins et al., 2006a), and LRGTS (Prins et al., 2007)) on three sets of instances. The results show that the proposed method outperforms the other algorithms and a majority of best-known solutions are improved.

Yu et al. (2010) proposed a simulated annealing based heuristic (SALRP) for solving a two-layer LRP with capacitated depots and vehicles. The heuristic features a special solution encoding scheme that integrates location and routing decisions in order to enlarge the search space so that better solutions can be found. The proposed SALRP heuristic is tested on three sets of well-known benchmark instances and the results are compared with other heuristics: the clustering based heuristic (CH) (Barreto et al., 2007), SA-ACS (Bouhafs et al., 2006), GRASP (Prins et al., 2006b), MA|PM (Prins et al., 2006a), LRGTS (Prins et al., 2007), and GAHLS (Duhamel et al., 2008), MSLS (Prins et al., 2004), and the two-phase tabu search (Tuzun & Burke, 1999). Computational results and comparisons indicate that the proposed SALRP algorithm performs better than the other heuristics on many instances.

Other heuristic methods developed for two-layer LRPs can be seen in Chien (1993), Srivastava (1993), Nagy and Salhi (1996a, 1996b), Lin et al. (2002), Prins et al. (2004), Albareda-Sambola et al. (2005), Melechovský et al. (2005), Prins et al. (2005), Wang et al. (2005), Bouhafs et al. (2006), Lin et al. (2006), Prins et al. (2006a), Yang and Li-Jun (2006), Caballero et al. (2007), Prodhon (2007), Duhamel et al. (2008), and Marinakis and Marinaki (2008b).

Multi-Layer LRPs

Some studies have been performed on multi-layer LRPs. In the following, studies performed on multi-layer LRPs are reviewed.

Jacobsen and Madsen (1980) and Madsen (1983) presented three heuristics to solve a three-layer LRP to design a newspaper distribution network. The network consists of one printing office, transit points, and sales points. In the network, newspapers are delivered from the printing office to transfer points and from there to sales points. The problem includes one location problem, location of transit points; one allocation problem, allocating sales points to transfer points; two routing problems, primary tours from the printing office to transfer points and secondary tours from transfer points to sales points; and one problem of sequencing primary tours. There are two types of vehicles: primary vehicles for primary tours and secondary (smaller) vehicles for secondary tours. The constraints included in the model are production rate of the printing office, capacitated primary vehicles, latest delivery time at each sales point, and secondary tour duration (time window) constraints. The three heuristics discussed are TTH (Tree-Tour Heuristic), ALA-SAV (Alternate Location Allocation – Savings method) heuristic, and SAV-DROP (Savings method – Drop

method) heuristic. They showed that the ALA-SAV and the SAV-DROP have better results but need to be merged and improved to obtain better solutions.

Perl (1983) and Perl and Daskin (1985) formulated a three-layer LRP with plants, warehouses, and customers but presented a heuristic method to solve a two-layer LRP with warehouses and customers. The formulation includes the location of warehouses, allocation of customers to warehouses, routing from warehouses to customers, and transportation from plants to warehouses.

Bookbinder and Reece (1988) formulated a three-layer LRP. The distribution network consists of plants, depots, and customers. Multi-type products are shipped from plants to depots and then from depots to customers. The problems considered are the location of depots, allocation of customers to depots, routing from depots to customers, and transportation from plants to depots. They decomposed the problem into three sub-problems (location-allocation problem, routing problem, and transportation problem) and presented a solution procedure that solves the sub-problems sequentially in an iterative manner.

Bruns and Klose (1996) developed a solution algorithm to solve a three-layer LRP including plants, depots, and customers. Depots are supplied by plants, and customers are served on tours starting from depots. Constraints are plant capacity, depot capacity, vehicle capacity, and maximum tour duration. The problems are: 1) location, locating depots; 2) allocation, allocating customers to depots; and 3) transportation, product flow from plants to depots. The location-allocation sub-problem is solved using a Lagrangean heuristic, which is based on the relaxation of the supply and capacity constraints. To solve the

routing sub-problem, tour construction heuristics along with tour improvement procedures are used.

Bruns et al. (2000) studied a parcel delivery system that is shown in Figure 2.1. In the parcel delivery system, parcels are sent from post offices (PO) to parcel processing centers (PPC) and from there to delivery bases (DB). The parcels are delivered to customers through PPCs or DBs. The problem can be viewed as a four-layer LRP. However, the authors reduced the problem to a two-layer LRP by using some assumptions, such as fixing the locations of PPCs and assigning each DB to a specific PPC. Furthermore, by grouping customers into customer zones and using routing cost estimations, the authors reduced the problem to a simple location problem, where routing costs are subsumed into the assignment costs. In fact, the sub-problems are locating DBs and allocating customer zones to PPCs and selected DBs. The authors solved the problem using a branch-and-bound algorithm.

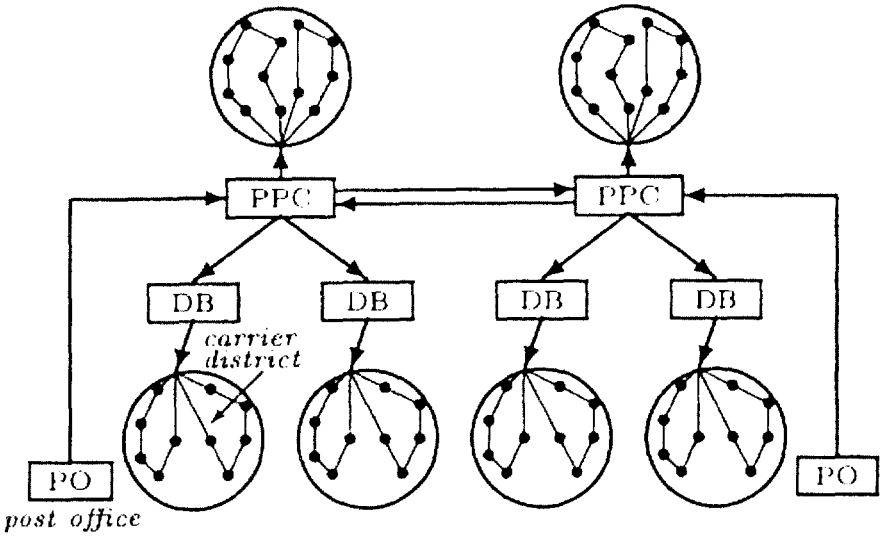

Figure 2.1. Structure of the parcel delivery system (Bruns et al., 2000).

Ambrosino and Scutella (2005) formulated a four-layer LRP with one plant, central depots (CD), transit points (TP), and customers. The plant sends goods to CDs. CDs transfer the goods to TPs and may serve big customers. And TPs deliver the goods to customers. The main problems to be solved are: 1) location at two levels, locating CDs and TPs; 2) allocation at two levels, allocating TPs and customers to CDs and allocating customers to TPs; 3) routing at two levels, routing between CDs, TPs, and customers starting from CDs and routing between TPs and customers starting from TPs; 4) the quantity of goods which must be shipped from the plant to CDs and from CDs to TPs. The distribution network is shown in Figure 2.2. They proposed two kinds of mathematical programming formulations and tried to solve them to optimality using CPLEX within a time limit of 25 hours for the small instances and several days for the largest instances. They reported the cost of the best integer solution found by CPLEX and the best lower bound provided by CPLEX (MIP bound). The optimum solution has been obtained only for one instance. For the others, the best integer solution is generally far from the best lower bound. These results suggest that the problem is very difficult to solve by using an exact method and the computational time is considerable even for small or mid-size instances. In order to close the gap, they suggested the study of some heuristic approaches for future work.

Aksen and Altinkemer (2008) proposed a three-layer LRP that consists of warehouses, stores, and customers (online and walk-in). As shown in Figure 2.3, goods are transferred from the warehouse to stores in direct shipments. Goods are sold to walk-in customers who directly go to open stores to receive their items. Also, goods are delivered to online customers on vehicle tours. The model combines store location, customer

allocation, and vehicle routing problems with the problem of transshipment of goods from warehouses to stores. Stores have unlimited storage capacity. The authors described a lagrangian relaxation-based solution method for the problem. They tested their method on a variety of test problems. The largest problem solved includes 1 warehouse, 4 stores, and 100 online customers. The CPU time for this instance is 17,822 seconds. The computational results show that the method is not suitable for large-size instances with 100 or more online customers as they require unfavorably long solution times.

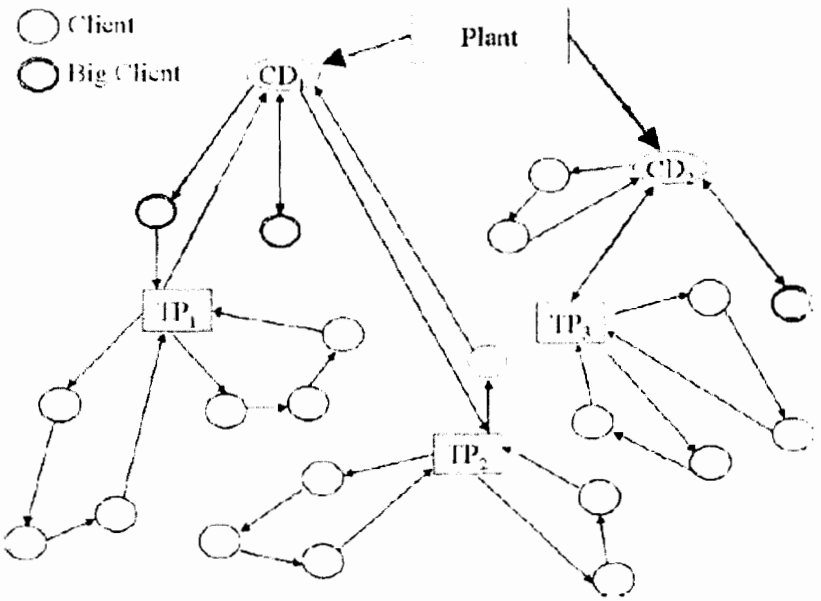

Figure 2.2. A four-layer LRP (Ambrosino & Scutella, 2005).

May and Tu (2008) modeled the pickup operation of international express service as a three-layer LRP with one warehouse (hub), transshipping points (TPs), and customers. The network is shown in Figure 2.4. In the first stage, customer orders are collected and delivered to TPs. In the second stage, collected freights are transferred from TPs to the warehouse. As shown in Figures 2.5 and 2.6, for the second stage two types of

transshipment models were introduced; Model (1): direct transshipment model and Model (2): routing transshipment model. In Model (1) freights are transported directly from each TP to the warehouse, while in Model (2) freights are transported from TPs to the warehouse through a loop (tour). The problems considered are: 1) locating TPs, 2) allocating customers to TPs, 3) routing between customers and TPs, and 4) routing between TPs and the warehouse. The authors presented two mathematical formulations for the models. They solved several randomly generated small-size problems with 4 TPs and 20 customers to optimality. However, the authors did not present any solution algorithm that can solve mid-size and large-size problems.

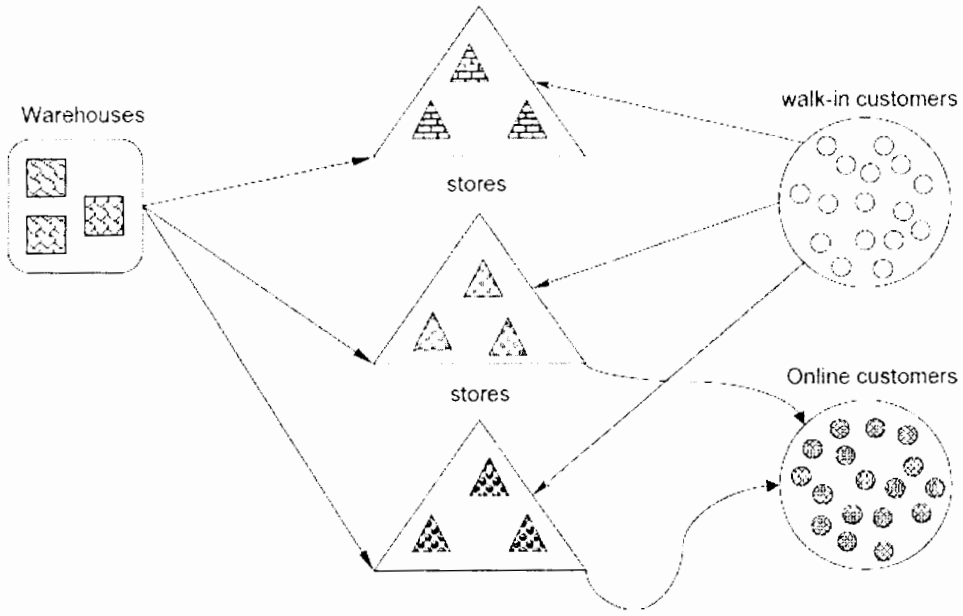

Figure 2.3. A three-layer LRP (Aksen & Altinkemer, 2008).

Lin and Lei (2009) formulated a three-layer LRP with two-level routing considerations. Main problems considered are: 1) the number and locations of distribution centers (DCs), 2) which big clients should be included in the first level routing (the routing

between plants, DCs, and big clients), 3) the first-level routing, and 4) the second-level routing between DCs and other clients not included in the first-level routing. The distribution network is shown in Figure 2.7. To solve the problem, they developed a hybrid genetic algorithm embedded with a routing heuristic. The first element is a genetic algorithm heuristic for locating DCs and the big clients included in the first level routing. The second is a simple routing heuristic that is used to find how to dispatch vehicles at the open DCs and how to dispatch vehicles at the plants. The simple routing heuristic is a cluster-base heuristic that consists of saving/insertion algorithms and tour improvement/exchange algorithms. The quality of the solution to a series of small test problems is evaluated (by comparison with the optimal solution solved using LINGO 9.0). In test problems for which exact solutions are available, the heuristic solution is within 1% of optimal. They also compared their algorithm to four LRP heuristics: TS of Tuzun and Burke (1999), GRASP of Prins et al. (2006b), MA|PM of Prins et al. (2006a), and LRGTS of Prins et al. (2007) on a set of instances designed by Tuzun and Burke (1999). The proposed solution procedure is able to find good solutions close to the best known results (the average gap is less than 3.4%).

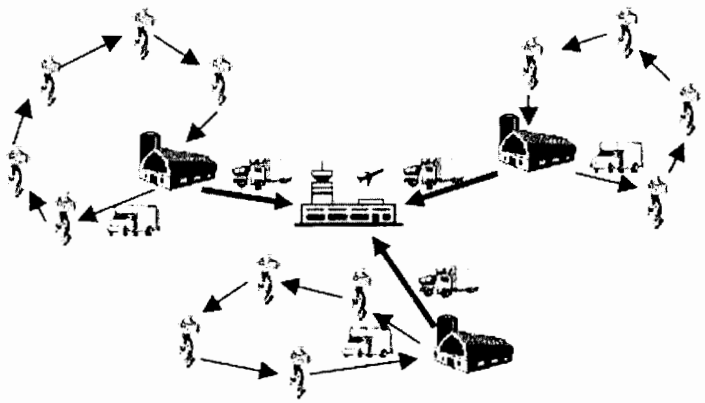

Figure 2.4. A three-layer LRP (May & Tu, 2008).

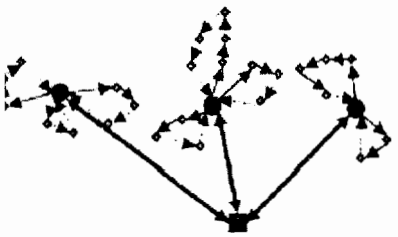

Figure 2.5. Model (1) (May & Tu, 2008).

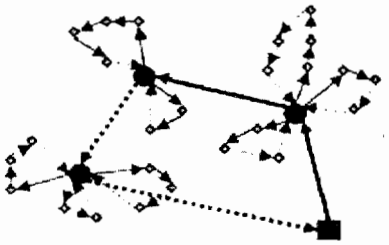

Figure 2.6. Model (2) (May & Tu, 2008).

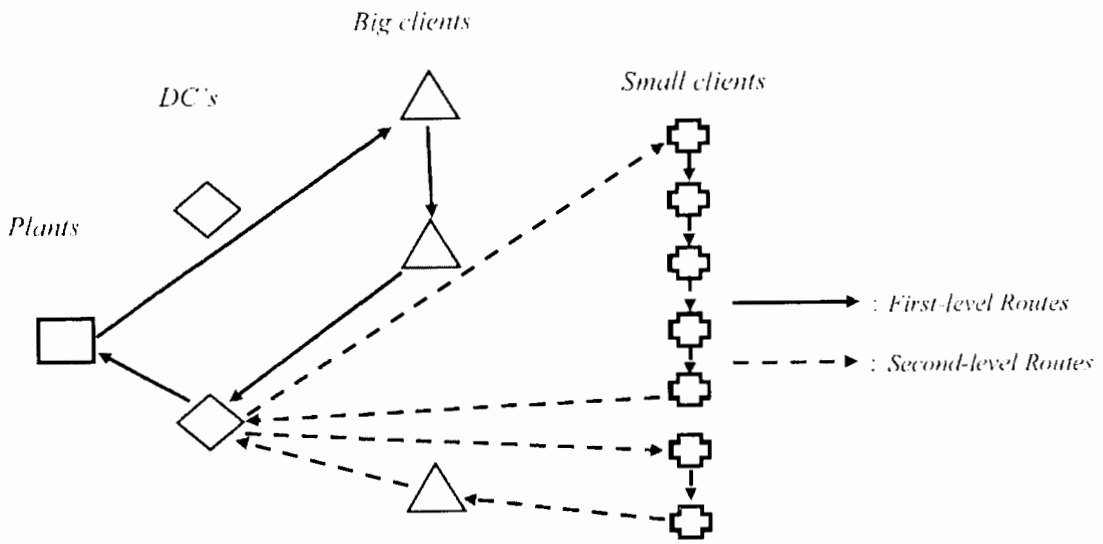

Figure 2.7. A three-layer LRP (Lin & Lei, 2009).

Ambrosino et al. (2009) addressed a special LRP with three layers: one central depot, regional depots, and customers. In the distribution network, customers are already partitioned into regions. In each region one regional depot is to be located. The central depot supplies general goods and the regional depots supply regional goods. The customers need both general and regional goods. In each region, routes are to be designed, and each route has to include both the central depot and the designated regional depot. In fact, each vehicle starts its tour at the central depot, goes to an open regional depot for loading the regional items, and then after having visited a certain number of customers in the associated region returns to the central depot. In Figures 2.8 and 2.9 the distribution network is visualized. In Figure 2.9 node 0 is the central depot and node 10 is a regional depot. The authors presented a mixed-integer linear programming model for the problem. They proposed a two-phase heuristic to solve the problem. The first phase determines a feasible solution by using a "location-allocation-routing" procedure based on a "cluster-first, route-second" method. The second phase improves the initial solution by using very large neighborhood search techniques, based on multi-exchange moves, for modifying the routing decisions, and using more classical relocation moves for improving the location aspects. The heuristic was tested on several randomly generated instances. On small and medium-size instances (2-4 regions, 4-32 candidate depots, and 20-60 customers), the heuristic approach is able to determine good quality solutions in a limited amount of time. As the authors have mentioned, the heuristic appears to be efficient only for small and medium-size instances. For large instances (4-6 regions, 16-90 candidate regional depots, and 80-420 customers) the heuristic seems to be not efficient as the computational time is very large (e.g. more than 18,000 seconds).

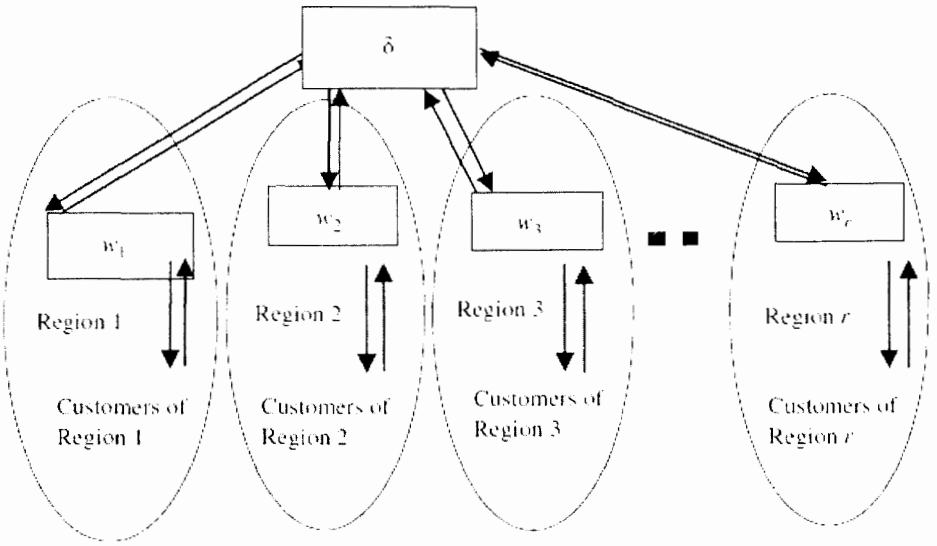

Figure 2.8. A three-layer LRP (Ambrosino et al., 2009).

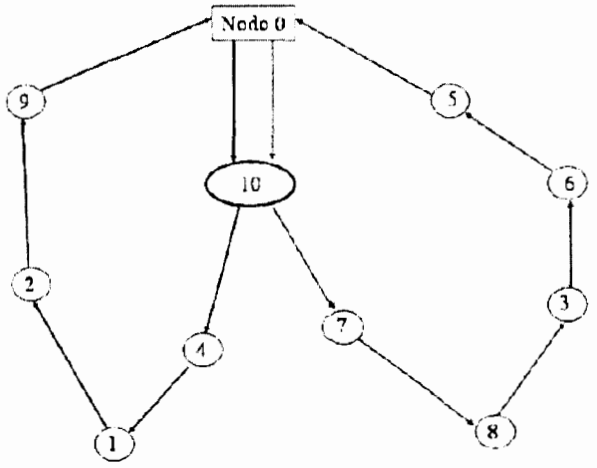

Figure 2.9. A three-layer LRP (Ambrosino et al., 2009).

Lee et al. (2010) presented a four-layer LRP which includes suppliers, manufacturers, distribution centers (DC), and customers. The study considers a supply chain in which suppliers send materials to manufacturers, manufacturers send products to DCs, and DCs send products to customers. The supply chain network is shown in Figure 2.10. The problems considered are: 1) location at two layers, locating manufacturers and

DCs; 2) allocation at two layers, allocating suppliers to manufacturers and allocating customers to DCs; 3) routing at two layers, routing between manufacturers and suppliers starting from manufacturers and routing between DCs and customers starting from DCs; and 4) transportation, transporting products from manufacturers to DCs. They presented a mixed integer programming model that considers a single item (product) and capacity limitation for suppliers, manufacturers, and DCs. Since the model can solve very small problems for optimality, they developed a heuristic algorithm based on LP-relaxation (relaxed binary variables). In their heuristic, they relaxed the binary constraints of the manufacturers and DCs' location decision variables by determining lower bounds for the numbers of manufacturers and DCs. Then they determined which manufacturers and DCs to open. Given the locations, they solved suppliers-manufacturers routing and DCs-customers routing problems. And then they solved the transportation problem between manufacturers and DCs with routing results. They solved the location, routing, and transportation problems sequentially in an iterative manner. The largest problems they solved include 30 suppliers, 10 potential manufacturers, 10 potential DCs, and 30 customers. The heuristic's computational time for 400 iterations of the heuristic varies from 843 to 1,845 seconds. Considering the size of the problems, it seems the heuristic needs a considerable amount of computational time for even mid-size problems and may not be applied for large-size problems.

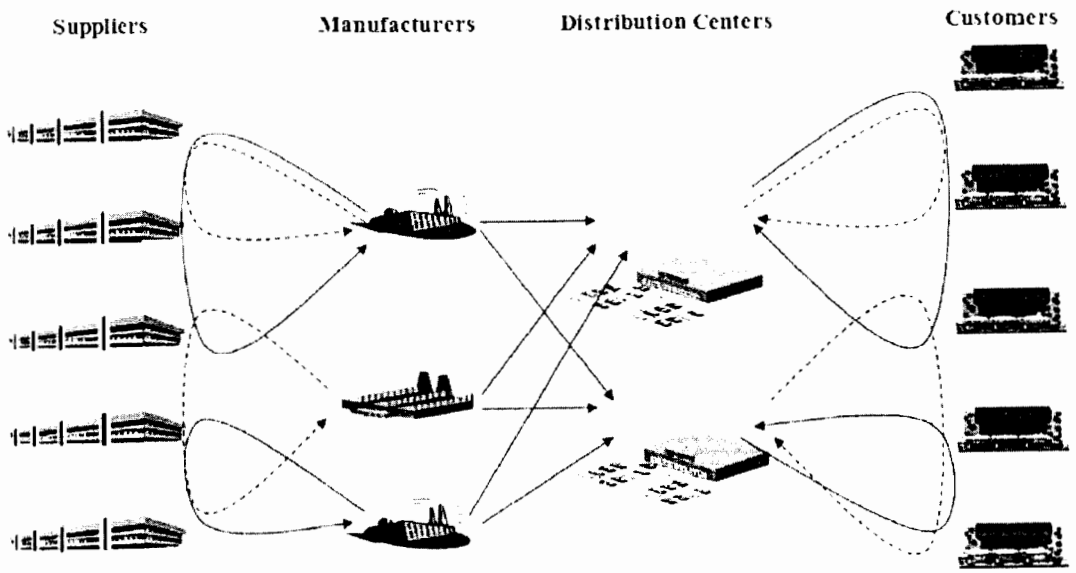

Figure 2.10. A four-layer LRP (Lee et al., 2010).

CHAPTER 3. MODELING

A Four-Layer LRP

As mentioned by Nagy and Salhi (2007), who conducted the most recent survey on LRP, one of the gaps in the LRP literature is modeling and solving multi-layer LRPs. The problem under consideration in this study is a complex four-layer LRP that has not been tackled yet. The four-layer LRP represents a multi-product distribution network consisting of plants, central depots, regional depots, and customers. The general characteristics of the network are as follows. To visualize the four-layer LRP, the graphic of the network is presented in Figure 3.1.

1) The network consists of plants (P), layer 1 (in green); central depots (CD), layer 2 (in purple); regional depots (RD), layer 3 (in blue); and customers (C), layer 4 (in orange).
2) Plants produce multiple-type products.
3) Products can be shipped from plants to other plants, CDs, RDs, or customers. From CDs, products can be shipped to other CDs, RDs, or customers. And from RDs, products are delivered to customers. Basically, customers' demands can be satisfied directly by plants, CDs, or RDs.
4) Each route between facilities (plants, CDs, and RDs) consists of only two facilities (tours are not allowed). To deliver products to customers, multiple customers can be visited in one route (tours are allowed). Also, multiple tours are allowed from each facility to deliver products to different groups of customers that are allocated to the

facility. To avoid complexity in Figure 3.1, one tour is shown from each facility to customers, but as mentioned above, multiple tours are allowed.

5) Products can be shipped within layer 1 from one plant to another plant since some products may not be produced in a certain plant or the plant may not have enough production capacity. Products can be shipped from plants to CDs and RDs directly or indirectly. CDs are distribution hubs and transshipment points, have larger capacity, and can be used to transship products to other depots. Basically, if a depot is far from a plant, it can be more economical to ship products to a CD and from there to the depot. This can avoid the increase in the travelling cost per mile due to the costs associated with overnight lodging of the driver and off-time hours when the driver does not work but gets paid overnight. Also, the limited service hours of the driver per day may limit the daily distance travelled. In addition, large trucks can be used to ship products to CDs and small trucks can be used to ship products to RDs, which is a more economical way as more truck capacity is used and the transportation cost per unit decreases. So, shipments between two facilities (plants, CDs, and RDs) cannot take place if the distance between the facilities is more than a maximum direct shipment distance. Products are not allowed to be shipped within layer 3 from one RD to another RD and this is what differentiates CDs from RDs. CDs are distribution hubs and transshipment points while RDs are not. Besides delivering products to customers, CDs mix different products obtained from plants and prepare and ship the requirements of other depots, especially RDs.

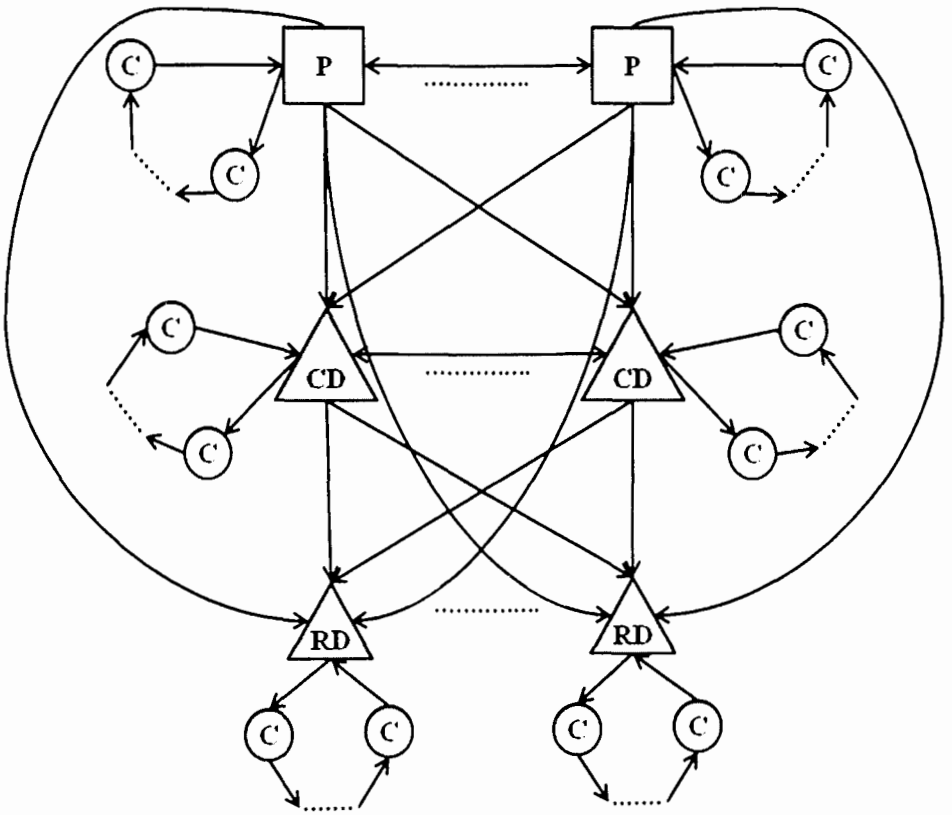

Figure 3.1. The graphic of the four-layer LRP.

Unlike Ambrosino and Scutella's (2005) study, this network is a multi-plant and multi-product network which represents realistic situations of distribution networks where multiple products are produced and shipped from multiple plants. Also, unlike Ambrosino and Scutella's (2005) and Lee, Moon, and Park's (2010) studies, in this network customers not only can be served by depots, but also can be served by plants, considering the fact that plants have warehouses attached to them and can serve nearby customers.

Basically, the presented LRP is a combination of location, allocation, vehicle routing, and transshipment problems. A transshipment problem allows shipment between supply points and between demand points, and it may also contain transshipment points

through which goods may be shipped on their way from a supply point to a demand point (Winston, 2003). The following is the list of the problems to be solved and the decisions to be made in this LRP:

1) Location problem at two layers (locating CDs and RDs)
   - Given a set of candidate sites for CDs, how many CDs are needed and where should they be located?
   - Given a set of candidate sites for RDs, how many RDs are needed and where should they be located?
2) Allocation problem at three layers (assigning customers to plants, CDs, and RDs)
   - Which customers should be served by each plant?
   - Which customers should be served by each CD?
   - Which customers should be served by each RD?
3) Transshipment problem
   - In each plant how many units of each product should be produced?
   - From each plant how many units of each product should be directly shipped to which plant, CD, or RD?
   - From each CD how many units of each product should be directly shipped to which CD or RD?
4) Vehicle routing problem at three layers (tours from plants, CDs, and RDs to customers)
   - To visit and serve the customers allocated to a plant, a CD, or an RD, how many tours should exist, which customers should be on each tour, and what is the best sequence of the customers on each tour?

The main assumptions and limitations of the model are as follows:

1) Plants are fixed, and their locations are known.
2) CDs and RDs are to be located in a set of candidate locations.
3) Customers' locations are known, and their demands are known.
4) Plants have limited production capacity for producing products.
5) CDs and RDs have limited space capacity.
6) Vehicles have limited capacity, and one type of vehicle is used to deliver products to customers.
7) Each customer's demand is less than the vehicle's capacity (less than truck load (LTL)).
8) The customer's demand is satisfied by only one vehicle (each customer is only on one tour).
9) Each tour starts and ends at the same facility.
10) Limitations on travel distances are considered. In delivering products to customers, lengths of tours are limited. Some reasons for this limitation are perishable items being transported, limited drivers' service hours per day/tour, avoiding overnight costs, and faster and more reliable delivery to customers. Also, shipments between two facilities (plants, CDs, and RDs) cannot take place if the distance between the facilities is more than a maximum direct shipment distance.

Mathematical Model

To solve the four-layer LRP and make the decisions mentioned in the previous section, the following mixed integer programming model is developed. The components of the model are as follows:

Sets:

P : set of plants

CD : set of candidate sites for CDs

RD : set of candidate sites for RDs

C : set of customers

T : set of tours

PR : set of products

Input parameters:

o_j : cost of operating a depot at site j

sc_m : cost per mile of direct shipment of one unit of product m between facilities (plants, CDs, and RDs)

td_{gh} : travelling distance between point g and point h

tc : travelling cost per mile of a vehicle on a tour

ft : fixed cost of a tour

tl : maximum allowable length of a tour

d_{im} : demand of customer i for product m

su_m : number of standard units per one unit of product m in terms of space needed

vc : vehicle capacity, number of standard units

dc_j : capacity of depot (CD, or RD) j, number of standard units

pc_{pm} : production capacity of plant p for product m

fd : maximum allowable distance to be travelled for direct shipment from one facility to another facility

Decision variables:

$X_{ghk} = 1$ if point g immediately precedes point h on tour k and 0 otherwise

$Y_{ij} = 1$ if customer i is served by facility j and 0 otherwise

$Z_j = 1$ if a depot is located at site j and 0 otherwise

U_{jlm} : Number of units of product m to be shipped from facility j to facility l

V_{pm} : Number of units of product m to be produced in plant p

Formulation:

Minimize

$$\sum_{j\in CD\cup RD} o_j Z_j + \sum_{m\in PR} \sum_{l\in P\cup CD\cup RD} \sum_{j\in P\cup CD} sc_m td_{jl} U_{jlm} + \sum_{k\in T} \sum_{h\in P\cup CD\cup RD\cup C} \sum_{g\in P\cup CD\cup RD\cup C} (tc) td_{gh} X_{ghk} + \sum_{k\in T} \sum_{h\in C} \sum_{g\in P\cup CD\cup RD} (ft) X_{ghk} \quad (1)$$

Subject to:

$$\sum_{k\in T} \sum_{h\in P\cup CD\cup RD\cup C} X_{ihk} = 1, \quad \forall i \in C \quad (2)$$

$$\sum_{g\in P\cup CD\cup RD\cup C} X_{ghk} - \sum_{g\in P\cup CD\cup RD\cup C} X_{hgk} = 0, \quad \forall h \in P\cup CD\cup RD\cup C, \ \forall k \in T \quad (3)$$

$$\sum_{k\in T} \sum_{h\in\{P\cup CD\cup RD\cup C\}-S} \sum_{g\in S} X_{ghk} \geq 1, \quad \forall S \subset P\cup CD\cup RD\cup C, \ P\cup CD\cup RD \subseteq S \quad (4)$$

$$\sum_{j\in P\cup CD\cup RD} \sum_{i\in C} X_{ijk} \leq 1, \quad \forall k \in T \quad (5)$$

$$\sum_{h\in P\cup CD\cup RD\cup C} X_{ihk} + \sum_{h\in P\cup CD\cup RD\cup C} X_{jhk} - Y_{ij} \le 1, \quad \forall i \in C,\ \forall j \in P\cup CD\cup RD,\ \forall k \in T \tag{6}$$

$$\sum_{h\in P\cup CD\cup RD\cup C} \sum_{g\in P\cup CD\cup RD\cup C} td_{gh} X_{ghk} \le tl, \quad \forall k \in T \tag{7}$$

$$\sum_{m\in PR} \sum_{h\in P\cup CD\cup RD\cup C} \sum_{i\in C} d_{im} su_m X_{ihk} \le vc, \quad \forall k \in T \tag{8}$$

$$V_{pm} = \sum_{l\in P\cup CD\cup RD} U_{plm} + \sum_{i\in C} d_{im} Y_{ip} - \sum_{l\in P} U_{lpm}, \quad \forall p \in P,\ \forall m \in PR \tag{9}$$

$$\sum_{j\in P\cup CD\cup RD} Y_{ij} = 1, \quad \forall i \in C \tag{10}$$

$$V_{pm} \le pc_{pm}, \quad \forall p \in P,\ \forall m \in PR \tag{11}$$

$$\sum_{l\in P\cup CD} U_{ljm} - \sum_{i\in C} d_{im} Y_{ij} - \sum_{l\in CD\cup RD} U_{jlm} = 0, \quad \forall j \in CD,\ \forall m \in PR \tag{12}$$

$$\sum_{l\in P\cup CD\cup RD} U_{ljm} - \sum_{i\in C} d_{im} Y_{ij} = 0, \quad \forall j \in RD,\ \forall m \in PR \tag{13}$$

$$\sum_{m\in PR} \sum_{i\in C} d_{im} su_m Y_{ij} + \sum_{l\in CD\cup RD} U_{jlm} - dc_j Z_j \le 0, \quad \forall j \in CD \tag{14}$$

$$\sum_{m\in PR} \sum_{i\in C} d_{im} su_m Y_{ij} - dc_j Z_j \le 0, \quad \forall j \in RD \tag{15}$$

$$td_{jl} U_{jlm} - (fd) U_{jlm} \le 0, \quad \forall j \in P\cup CD,\ \forall l \in P\cup CD\cup RD,\ \forall m \in PR \tag{16}$$

$$X_{ghk} \in \{0,1\}, \quad \forall g,h \in P\cup CD\cup RD\cup C,\ \forall k \in T \tag{17}$$

$$Y_{ij} \in \{0,1\}, \quad \forall i \in C,\ \forall j \in P\cup CD\cup RD \tag{18}$$

$$Z_j \in \{0,1\}, \quad \forall j \in P\cup CD\cup RD \tag{19}$$

$$U_{jlm} \ge 0, \quad \forall j,l \in P\cup CD\cup RD,\ \forall m \in PR \tag{20}$$

$$V_{pm} \geq 0, \quad \forall p \in P, \ \forall m \in PR \tag{21}$$

The objective function is the distribution cost including depot cost, shipment (between facilities) cost, and delivery (to customers) cost (fixed and variable costs of tours). Constraints (2) ensure that each customer is on only one tour (is served by only one vehicle). Constraints (3) are the tour continuity constraints which imply that every point that is entered by a vehicle should be left by the same vehicle. Constraints (4) require that every tour be connected to a facility; they ensure that there is at least one connection from the set of facilities and any customer(s) to the rest of the customers. Constraints (5) state that each tour cannot be operated from multiple facilities. Constraints (6) link the allocation and routing problems; they specify that a customer can be allocated to a facility only if there is a tour from that facility going through that customer. Constraints (7) limit the length of each tour. Constraints (8) guarantee that the space needed for the demand of customers in each delivery does not exceed the capacity of a vehicle. Constraints (9) calculate the number of each product that has to be produced in each plant. Constraints (10) assure that each customer is assigned to only one facility; although this set of constraints is not seen in LRP models, during modeling and solving some problems it was seen that because of having constraints (9) in this particular LRP, it is necessary to have this set of constraints. Constraints (11) state that the number of products to be produced in each plant cannot exceed the production capacity of the plant. Constraints (12) and (13) specify that the flow into a depot is equal to the flow out of the depot. Constraints (14) and (15) state that the space needed for a depot should not exceed the depot's capacity. These constraints also assure that a depot cannot be opened unless it is used for delivering products to customers or transshipping products to other depots. Constraints (16) guarantee that a

direct shipment between two facilities takes place only if the distance between the facilities is less than a maximum allowable distance. Constraints (17), (18), and (19) are binary integer constraints and constraints (20) and (21) are non-negativity constraints for decision variables.

To test the model, a small-size problem has been formulated in GAMS software and solved for optimality. The GAMS model (including values of input parameters) is presented in Appendix A. The optimal network and solution are shown in Figure 3.2 and Table 3.1. As shown in Figure 3.2, the distribution network consists of two plants (points 1 and 2), two CDs (points 3 and 4), two RDs (points 5 and 6), and four customers (points 7, 8, 9, and 10) and two types of products are produced. In Table 3.1, the optimal values of the decision variables are presented. The total optimal cost is 24700. The model constraints have been checked manually and it has been verified that they are satisfied.

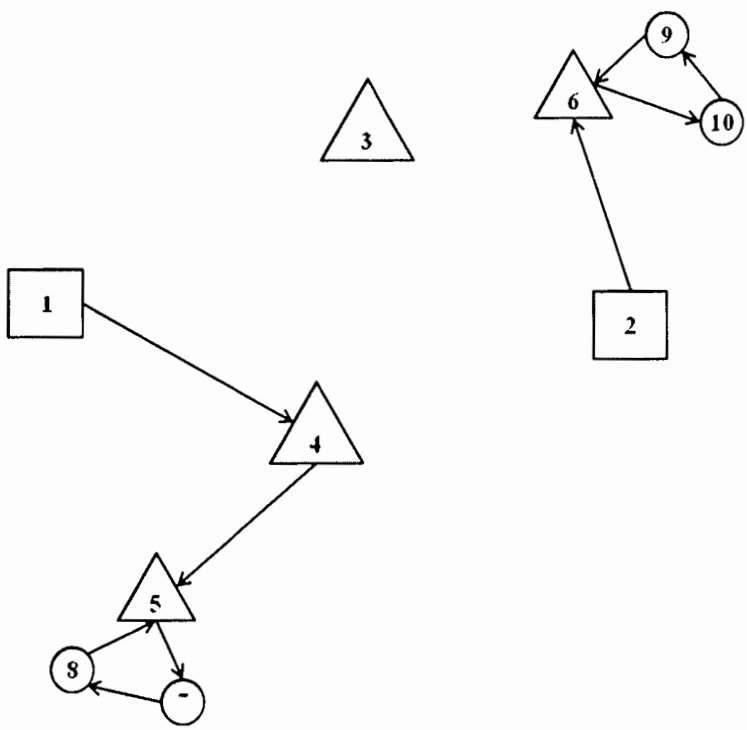

Figure 3.2. The optimal network.

Table 3.1. The optimal solution.

Decision / Variable	Variable Values
Tour routing (X_{ghk})	$X_{6(10)1} = X_{(10)91} = X_{961} = 1$ $X_{572} = X_{782} = X_{852} = 1$
Customer allocation (Y_{ij})	$Y_{75} = Y_{85} = 1$ $Y_{96} = Y_{(10)6} = 1$
Depot location (Z_j)	$Z_4 = Z_5 = Z_6 = 1$
Transshipment (V_{pm} and U_{jlm})	$V_{11} = 40$, $V_{12} = 15$ $V_{21} = 50$, $V_{22} = 25$ $U_{141} = 40$, $U_{142} = 15$ $U_{451} = 40$, $U_{452} = 15$ $U_{261} = 50$, $U_{262} = 25$

The above four-layer LRP and mathematical model are presented and discussed in Hamidi et al. (2011) and Hamidi et al. (2012a).

CHAPTER 4. METAHEURISTIC SOLUTIONS

The presented four-layer LRP is one of the complex LRPs, and the mathematical model presented in the previous chapter can solve only small problems in a reasonable amount of time. As mentioned earlier, LRP is an NP-hard problem and contains too many variables and constraints to be solved using an exact algorithm. Besides, multi-layer and multi-product LRP is even more complex as it deals with the decisions at multiple layers of a distribution network where multiple products are transported within and between layers of the network. The reasons for complexity are as follows:

- Dealing with two types of depots (CDs and RDs) to locate and solving the location problem while the location decisions affect not only allocation and routing decisions but also transshipment decisions.
- Dealing with three types of facilities (plants, CDs, and RDs) for allocating customers and considering the effects of allocation decisions on location, transshipment, and routing cost. For example, assigning a customer which is far from plants to a given plant in order to avoid opening a closed depot decreases the location and transshipment costs but increases the routing cost.
- Having multiple plants producing multiple types of products.
- Adding transshipment problem to LRP: Transshipment cost is included in the total cost (objective function) and so in the solution process the effects of transshipment decisions on other components of the total cost (location cost and routing cost) and their interactions should be considered. Transshipping products that can be

performed through multiple facilities at three layers (plants, CDs, and RDs) deals with many possibilities that are interrelated with location, allocation, and routing decisions. For example, if an RD is far from plants and is to be opened to serve customers, at least one CD has to be opened to transship the requirements of the customers of the RD regardless of whether or not some customers have been assigned to the CD.

- Taking into consideration the following constraints in addition to the regular constraints (depot capacity and vehicle capacity) that typical LRPs consider:
  1) Plant production capacity
  2) CD capacity for transshipping products to other depots
  3) Tour length limitation
  4) Direct shipment (between facilities) distance limitation

To solve large problems effectively, generating high-quality and near optimal solutions during a reasonable amount of time, metaheuristic algorithms should be applied. To solve different versions of the problem and decrease the complexity of the solution algorithm for less complicated cases, the following two scenarios are introduced. The first scenario decreases the complexity of the algorithm for less complicated cases.

- **Scenario 1:** In this scenario, the first two constraints mentioned above are relaxed. Each plant produces all types of products. Plants have enough production capacity or are able to increase their production capacity. Also, to transship products to other depots, CDs have enough capacity or once they receive products, immediately or after a short amount of time they ship products to other depots. In this scenario, in

addition to the regular constraints (depot capacity and vehicle capacity) that typical LRPs consider, the tour length constraint and direct shipment distance limitation are taken into consideration. This scenario covers the cases in which the relaxed constraints do not exist. Even for some problems in which those constraints exist, the first scenario may work (in case the final solution does not violate the relaxed constraints). This way, the problem can be solved by using a less complex and faster solution algorithm.

- **Scenario 2:** In the second scenario, the constraints that were relaxed in the first scenario will be added to the problem and so all of the constraints are considered. Basically, each plant does not necessarily produce all types of products or plants have limited production capacity for products. Also, CDs have limited capacity for transshipping products to other depots. The second scenario is more complicated than the first scenario as it considers two more constraints.

Scenario 1

In this section a metaheuristic that can solve large problems effectively, generate high-quality and near optimal solutions during a reasonable amount of time, is presented. As stated by Osman and Laporte (1996), "a metaheuristic is formally defined as an iterative generation process which guides a subordinate heuristic by combining intelligently different concepts for exploring and exploiting the search space, learning strategies are used to structure information in order to find efficiently near-optimal solutions."

In general, LRP heuristic solutions are composed of two phases: 1) generating an initial feasible solution and 2) improving the initial solution toward a final solution close to

the optimal solution. A good initial solution helps the second phase (improvement phase) to reach the final solution faster and find a very good (optimal or very close to optimal) final solution. In other words, a good initial solution increases the potential to improve the solution faster and more efficiently. An effective improvement phase is the one that in each step of improvement, simultaneously considers the interrelated sub-problems of the problem (their constraints and their interactive contributions on the objective function and quality of the solution). Considering the sub-problems simultaneously is also a factor that can be taken into account while the initial solution is being generated. However, the more sub-problems and the more complex the interrelations between them, the more difficult it becomes to consider the sub-problems simultaneously. Consequently, the heuristics usually have a sequential structure in which in each step of the improvement phase (whether or not the steps are iterative), only one or two sub-problems are taken into consideration. Also, as the number of layers of LRP increases, the interrelations between the sub-problems become more complex and considering the sub-problems simultaneously more difficult. So, generating a good initial solution and developing effective improvement procedures become more difficult as the number of the layers of LRP and the number of sub-problems increase.

The metaheuristic solution for Scenario 1 uses a combination of GRASP and tabu search metaheuristics along with local search techniques to solve the problem.

GRASP Metaheuristic

GRASP (Greedy Randomized Adaptive Search Procedure) is a metaheuristic for solving difficult combinatorial optimization problems (Feo & Resende, 1995). GRASP is a

multi-start or iterative process, with each GRASP iteration consists of two phases: a construction phase and a local search phase (Feo & Resende, 1995). The construction phase builds a feasible solution, whose neighborhood is investigated until a local optimal is found during the local search phase (Resende & Ribeiro, 2003). The construction phase and local search phase take place iteratively and terminate when some termination criterion, such as maximum number of iterations have occurred or solution sought has been found, is satisfied (Feo & Resende, 1995). The best overall solution is the final solution. The pseudo-code in Figure 4.1 illustrates a GRASP procedure.

```
procedure GRASP(Max_Iterations,Seed)
1    Read_Input();
2    for k = 1,...,Max_Iterations do
3        Solution ← Greedy_Randomized_Construction(Seed);
4        Solution ← Local_Search(Solution);
5        Update_Solution(Solution,Best_Solution);
6    end;
7    return Best_Solution;
end GRASP.
```

Figure 4.1. Pseudo-code of GRASP (Resende & Ribeiro, 2003).

As illustrated in Feo and Resende (1995), in the construction phase, a feasible solution is iteratively constructed, one element at a time. At each construction iteration, the choice of the next element to be added is determined by ordering all elements in a candidate list with respect to a greedy function. This function measures the (myopic) benefit of selecting each element. The heuristic is adaptive because the benefits associated with every element are updated at each iteration of the construction phase to reflect the changes brought on by the selection of the previous element. The list of best candidates is called the restricted candidate list (RCL). The probabilistic component of a GRASP is

characterized by randomly choosing one of the best candidates in the list, but not necessarily the top candidate. Figure 4.2 illustrates the construction phase with its pseudo-code.

```
procedure Greedy_Randomized_Construction(Seed)
1    Solution ← ∅;
2    Evaluate the incremental costs of the candidate elements;
3    while Solution is not a complete solution do
4         Build the restricted candidate list (RCL);
5         Select an element s from the RCL at random;
6         Solution ← Solution ∪ {s};
7         Reevaluate the incremental costs;
8    end;
9    return Solution;
end Greedy_Randomized_Construction.
```

Figure 4.2. Pseudo-code of the construction phase (Resende & Ribeiro, 2003).

As described in Feo and Resende (1995), the solutions generated by a GRASP construction are not guaranteed to be locally optimal with respect to simple neighborhood definitions. Hence, it is almost always beneficial to apply a local search to attempt to improve each constructed solution. A local search algorithm works in an iterative fashion by successively replacing the current solution by a better solution in the neighborhood of the current solution. It terminates when no better solution is found in the neighborhood. The pseudo-code of a basic local search algorithm starting from the solution constructed in the construction phase and using a neighborhood N is given in Figure 4.3.

By performing a simulation experiment and referring to several applications of GRASP, Feo and Resende (1995) have shown that GRASP can find very good solution, often optimal, in a limited number of iterations and short amount of time. GRASP

outperforms enumeration approach in which all combinations of solutions are evaluated; in combinatorial optimization problems, the number of solutions grows exponentially with the size of problem and evaluating all of the solutions is not possible in a reasonable amount of time (Feo & Resende, 1995). GRASP also outperforms the randomized solution approach in which a sample of solutions are randomly generated; the randomized solution approach doesn't take advantage of the greedy function and restricted candidate list and as a result the average quality of solutions is much worse than that of GRASP (Feo & Resende, 1995 and Resende & Ribeiro, 2003). In fact, GRASP contributes enormously to our ability to empirically find good solutions to otherwise unsolved instances of practical combinatorial optimization problems (Feo & Resende, 1995).

```
procedure Local_Search(Solution)
1    while Solution is not locally optimal do
2         Find s' ∈ N(Solution) with f(s') < f(Solution);
3         Solution ← s';
4    end;
5    return Solution;
end Local_Search.
```

Figure 4.3. Pseudo-code of the local search phase (Resende & Ribeiro, 2003).

Tabu Search Metaheuristic

As illustrated in Glover and Laguna (1997), tabu search is a metaheuristic that guides a local heuristic search procedure to explore the solution space beyond local optimality. The local search procedure is a search that uses an operation called move to define the neighborhood of any given solution. One of the main components of tabu search

is its use of adaptive memory, which creates a more flexible search behavior. Memory-based strategies are therefore the hallmark of tabu search approaches. Tabu search is concerned with imposing restrictions to guide a search process to negotiate otherwise difficult regions. These restrictions operate in several forms, both by direct exclusion of search alternatives classed as forbidden and also by translation into modified evaluations and probability of selection. Restrictions are imposed or created by making reference to memory structures that are designed for the specific purpose. Tabu search is based on the premise that problem solving, in order to qualify as intelligent, must incorporate adaptive memory and responsive exploration. The adaptive memory feature of tabu search allows the implementation of procedures that are capable of searching the solution space economically and effectively. Since local choices are guided by information collected during the search, tabu search contrasts with memoryless designs that heavily rely on semirandom processes that implement a form of sampling. Examples of memoryless methods include semigreedy heuristics and the prominent "genetic" and "annealing" approaches inspired by metaphors of physics and biology. Adaptive memory also contrasts with rigid memory designs typical of branch and bound strategies. The emphasis on responsive exploration in tabu search, whether in a deterministic or probabilistic implementation, derives from the supposition that a bad strategic choice can yield more information than a good random choice. In a system that uses memory, a bad choice based on strategy can provide useful clues about how the strategy may profitably be changed. Responsive exploration integrates the basic principles of intelligent search, i.e., exploiting good solution features while exploring new promising regions. Tabu search is concerned

with finding new and more effective ways of taking advantage of the mechanisms associated with both adaptive memory and responsive exploration.

Tabu search in its simplest form uses a short term memory strategy to intelligently direct its search away from neighborhoods already considered (Feo & Resende, 1995). Medium and long term memory strategies are respectively used in tabu search to allow for search intensification and diversification with regard to a known set of promising solutions (Feo & Resende, 1995).

Probabilistic tabu search select moves according to probabilities based on the status and evaluations assigned to these moves and can be summarized as follows (Glover & Laguna, 1997):

a) Create move evaluations that include reference to tabu status and other relevant biases from tabu search strategies, using penalties and inducements to modify an underlying decision criterion.
b) Map these evaluations into positive weights, to obtain probabilities by dividing by the sum of weights. The highest evaluations receive weights that disproportionately favor their selection.

As mentioned in Glover and Laguna (1997), two highly important components of tabu search are intensification and diversification strategies. Intensification strategies are based on modifying choice rules to encourage movement combinations and solution features historically found effective. They may also initiate a return to attractive regions to search them more thoroughly. The diversification strategies on the other hand encourage the search process to examine unvisited regions and to generate solutions that differ in

various significant ways from those seen before. Some diversification strategies are designed to prevent searching processes form cycling, i.e., from endlessly executing the same sequence of moves or revisiting the same set of solutions. Diversification strategies are particularly helpful when better solutions can be reached only by crossing barriers or humps in the solution space topology.

A Hybrid GRASP-Tabu Search Metaheuristic for Scenario 1

The metaheuristic developed to solve the four-layer LRP decomposes the problem into two sub-problems, a location-allocation-transshipment problem and a routing problem. Two metaheuristics of GRASP and tabu search are combined to solve the first sub-problem in which the routing cost is considered implicitly. The Clarke-Wright Savings algorithm and a node ejection chains algorithm are also used to solve the routing problem. Hence, the solution algorithm consists of two procedures: 1) location-allocation-transshipment procedure and 2) routing procedure. The main blocks of the algorithm are shown in Figure 4.4. Procedure 1 locates open depots, allocates customers to facilities, and constructs transshipment paths. Given the solution of Procedure 1, Procedure 2 constructs tours from facilities to customers. The details of the two procedures are as follows.

Procedure 1: Location-Allocation-Transshipment Procedure

As mentioned above, Procedure 1 locates open depots, allocates customers to facilities, and constructs transshipment paths. In this procedure, in each round of i, a GRASP generates n solutions in n iterations. In each iteration of j, the construction phase generates a greedy randomized solution and then the local search improves the solution and finds the local optimal solution. The components of this procedure are as follows.

```
Procedure 1: Location-Allocation-Transshipment
    for i = 1 to m rounds
        for j = 1 to n iterations
            Do construction phase of GRASP;
            Do local search phase of GRASP;
        end
        Update the list of elite solutions;
        Do diversification and intensification using probabilistic tabu search;
    end
    Select the best solution;
End of Procedure 1

Procedure 2: Routing
    Do Clarke-Wright Savings algorithm;
    Do Ejection Chain;
End of Procedure 2
```

Figure 4.4. Main blocks of the hybrid GRASP-tabu search metaheuristic.

Construction Phase of GRASP

In each iteration of the GRASP, the following steps generate a greedy randomized solution. A facility can be a plant, a CD, or an RD. Also, a depot can be a CD or an RD.

Step 1. For each plant, allocate the customers whose distances from the plant are less than or equal to the coverage radius. The coverage radius is the maximum desirable distance of a customer from a facility to which the customer can be allocated. From the practical point of view, it is not desirable to assign distant customers to the facility because multiple customers are to be served in each tour and the tour length is limited. So, this radius can be a fraction (less than ½) of the maximum tour length, e.g. one fourth of the maximum tour length, to make sure that multiple customers are served in one tour.

Step 2. Create the main list. Add all of the plants to the main list; the main list includes all plants and open CDs which will be added to the list in following steps. Select the first facility in the main list.

Step 3. Create the restricted candidate list (RCL) for the selected facility. The RCL for the facility is the list of depots with the following characteristics:

- The depot is unopened.
- The nearest customer to the depot has not been assigned to any facility.
- The depot can directly be connected to the selected facility; its distance to the facility is less than or equal to the maximum direct shipment distance.
- If the selected facility is a CD, only depots that cannot directly be connected to any plant can be in the RCL.

Step 4. From the RCL of the selected facility, randomly select a depot and do the following:

- Open the depot.
- Construct a connection from the facility to the depot; this is a transshipment path.
- If the depot is a CD, add it to the end of the main list.
- Assign the near unassigned customers to the depot until the remaining capacity of the depot is less than the customer's demand or the distance between the depot and the customer is more than the coverage radius; sort the unassigned customers based on their distances to the depot and start from the nearest customer.

Step 5. If all customers are assigned, finish the construction phase. Select the next facility in the main list and update its RCL. If the RCL is not empty, go to Step 4. Otherwise, restart Step 5. If the main list is exhausted, restart Step 5 (in this case the next facility is the

first facility in the main list). If the RCLs of all facilities in the main list are empty and at least one customer has not been assigned, go to Step 6.

Step 6. For an unassigned customer, find the nearest closed depot and open it. Assign the near unassigned customers to the depot until the remaining capacity of the depot is less than the customer's demand or the distance between the depot and the customer is more than the coverage radius; sort the unassigned customers based on their distances to the depot and start from the nearest customer. Restart this step until all customers are assigned. To construct transshipment paths for the depots opened in this step, go to Step 7.

Step 7. If the depot can be connected to a plant, connect it to the plant. If the depot can be connected to an open CD, connect it to the open CD. Otherwise, connect the depot to the closest closed CD and open the CD. If the open CD is already directly or indirectly connected to a plant, use the same transshipment paths to connect the depot to the plant. If the open CD is directly or indirectly not connected to a plant, restart and do Step 7 for the CD. Continue this step until all opened depots are directly or indirectly connected to plants.

In the GRASP presented above, to generate a greedy feasible solution, the four sub-problems described previously (location, allocation, transshipment, and routing problems) are considered simultaneously, and all of the constraints are taken into account. As seen in the above steps, facilities are categorized into three categories: Category 1: plants, Category 2: depots that can be connected to at least one plant (the distance between the depot and the plant is less than the maximum direct shipment distance), and Category 3: depots that cannot directly be connected to any plant. To decrease the transshipment cost, as a result of the order of facilities in the main list, for a facility to be selected, generally

category 1 facilities have priority over category 2 facilities and category 2 facilities have priority over category 3 facilities. Also, to decrease the transshipment cost, the algorithm tries not to connect facilities that are in category 2. To decrease the location cost, the algorithm tries to assign as many customer as possible to a selected depot in order to use the maximum depot capacity and to avoid opening another depot. To decrease the routing cost, the nearest customers are assigned to each depot. Moreover, because the depot cannot be included in the RCL if the nearest customer to the depot has been assigned to any facility, the routing cost and the number of opened depots decrease, which leads to decrease in the location cost.

After generating each solution, the total cost of the solution is calculated. The total cost includes the location cost, transshipment cost, and estimated routing cost. Since the tours are not constructed in this procedure, an estimated routing cost based on the allocation of customers to facilities is used. The routing cost includes the fixed cost and the variable cost. The fixed cost is the fixed cost per tour and the variable cost is basically the travelling cost. To estimate the travelling cost, the travelling distance or tour lengths should be estimated. A regression formula for route lengths per depot introduced by Christofides and Eilon (1969) and later extended by Daganzo (1984) and Stokx and Tilanus (1991) is shown in Equation (1) (Nagy & Salhi, 1996):

$$T = \frac{aD}{c} + \frac{bD}{\sqrt{n}} \qquad (1)$$

where T is the estimated total tour length, D is the sum of direct distances from the depot to all its assigned customers, n is the total number of customers assigned to the depot, c is the

average number of stops on one tour, and a and b are regression coefficient. The coefficients of $a = 1.8$ and $b = 1.8$ are suggested by Daganzo (1984).

In Procedure 1 of the presented metaheuristic, the routing cost, including fixed cost and variable cost, is calculated for each solution using the formula shown in Equation (2):

$$\textit{Estimated Routing Cost} = \sum_{j \in P \cup CD \cup RD} ft(nt_j) + \sum_{j \in P \cup CD \cup RD} tc\left(\frac{1.8D_j}{c_j} + \frac{1.8D_j}{\sqrt{n_j}}\right) \qquad (2)$$

where j represents facilities (plants, CDs, and RDs), ft is the fixed cost per tour, nt_j is the estimated number of tours for facility j, tc is travelling cost per mile of a vehicle on a tour, D_j is the sum of direct distances from facility j to customers assigned to facility j, c_j is the average number of stops on one tour, and n_j is the total number of customers assigned to facility j. The estimated number of tours for facility j (nt_j) is the ratio of $\frac{\hat{d}_j}{vc}$, which is rounded to the nearest integer larger than the ratio, where $\hat{d}_j$ is the sum of demands of customers assigned to facility j and vc is the vehicle capacity. The average number of stops on one tour (c_j) is basically the average number of customers on one tour and is the ratio of $\frac{vc}{\bar{d}_j}$ where $\bar{d}_j$ is the average demand of customers assigned to facility j. During the process of developing a heuristic algorithm for a two-layer LRP in Sajjadi et al. (2011), we found that this routing cost estimation method provides a more accurate estimate and leads to a decrease in cost compared to the simple method of the sum of direct distances of customers from depots.

Local Search of the GRASP

This phase improves the solution developed in the construction phase and finds the local optimal solution. As shown in Figure 4.5, the local search has three sub-procedures

which are attempted sequentially and iteratively until no improvement is achieved. The sub-procedures are described below.

```
while "Improvement is achieved"
    Do Transshipment Path Swap;
    Do Customer Reallocation;
    Do Depot Drop;
end
```

Figure 4.5. Local Search procedure of the GRASP (Scenario 1).

Transshipment Path Swap: For each open depot that is connected to a plant, if the depot is not connected to the nearest plant, disconnect the current connection and connect the depot to the nearest plant. For each open depot that is not directly connected to a plant (receives products from a CD, called CD 1), calculate the total transshipment distance from the depot to the designated plant, e.g. if products are transported to a CD and then to the depot, the total transshipment path is the sum of the distance between the plant and the CD and the distance between the CD and the depot. If the depot can be connected directly to a plant and the distance between the depot and that plant is less than the current transshipment path, disconnect the depot from CD 1 and connect it to the plant. Also, if there is an open CD, called CD 2, through which the transshipment path decreases, disconnect the depot from CD 1 and connect it to CD 2.

Customer Reallocation: For each customer, if the customer is assigned to depot A and depot A is not the nearest open depot to the customer, if the nearest open depot has a shorter transshipment path, direct or indirect, to a plant and has enough remaining capacity, change the allocation of the customer and assign it to the nearest open depot.

Depot Drop: For each open RD, if no customer-has been assigned to the RD, close it. For each open CD, if no customer has been assigned to the CD and the CD is not a transshipment point, through which products are not transported to other depots, close it.

The above sub-procedures have different impacts on the total cost. "Transshipment Path Swap" reduces the transshipment cost. "Customer Reallocation" reduces the sum of transshipment and routing costs. And "Depot Drop" reduces the location cost.

As shown in Figure 4.4, in Procedure 1 each round of the GRSAP includes n iterations and so generates n greedy randomized solutions. After each round, the list of elite solutions is updated; elite solutions are a limited number of the best solutions (with minimum costs) obtained so far.

Diversification and Intensification Using Probabilistic Tabu Search

To perform each round of the GRASP, three strategies of unbiased, intensification, and diversification are applied alternately. The intensification and diversification strategies in this procedure are designed in a probabilistic tabu search framework. The GRASP is memory-less and the tabu search adds a memory mechanism to the GRASP.

The intensification strategy generates solutions that are similar to the generated elite solutions by favoring the most frequently selected depots existing in the elite solutions. This strategy more thoroughly searches the most promising regions of the solution space and tries to find better solutions. To perform the intensification strategy, a short-term frequency-based memory is used. This memory consists of the number of times each depot is selected in the current elite solutions. When the intensification strategy is applied, in the construction phase of the GRASP (Step 4), the depots with a higher frequency are selected

with a higher probability than the depots with a lower frequency. In fact, the selection probability is proportional to the frequency and is obtained by Equation (3):

$$P_{j1} = \frac{freq_{j1}}{\sum_{j \in RCL} freq_j} \tag{3}$$

where P_{j1} is the selection probability of depot $j1$, and $freq_{j1}$ is the number of times depot $j1$ is selected in the current elite solutions.

The diversification strategy generates solutions that are different from generated solutions by favoring the least frequently selected depots. This strategy avoids solutions from being trapped in a local optimal and tries to search the areas of the solution space that have not been searched. To perform the diversification strategy, a long-term frequency-based memory is used. This memory consists of the number of times each depot is selected in all of the solutions generated so far. When the diversification strategy is applied, in the construction phase of the GRASP (Step 4), the depots with a lower frequency are selected with a higher probability than the depots with a higher frequency. In fact, the selection probability is inversely proportional to the frequency and is obtained by Equation (4):

$$P_{j1} = \frac{1/freql_{j1}}{\sum_{j \in RCL} (1/freql_j)} \tag{4}$$

where $freql_{j1}$ is the number of times depot $j1$ is selected in all of the solutions obtained so far.

The unbiased strategy is the strategy in which none of the intensification and diversification strategies is applied. It is basically a memory-less strategy which gives the procedure the opportunity to produce unbiased and more random solutions.

After *m* rounds of the GRASP, the best solution is selected. This solution includes location, allocation, and transshipment decisions.

Procedure 2: Routing Procedure

Given the solution of Procedure 1, Procedure 2 constructs tours from facilities to customers. Procedure 2 consists of one constructive algorithm that constructs tours from facilities to customers and one improvement algorithm that improves the tours. The Clarke-Wright Savings algorithm is used in the construction phase, and a node ejection chains algorithm is applied in the improvement phase.

The Clarke-Wright Savings Algorithm

To construct tours, the well-known Clarke-Wright Savings algorithm introduced by Clarke and Wright (1964) is used. Larsen and Odoni (2007) have summarized the steps of this algorithm.

Step 1. Calculate the savings $s(i, j) = td(D, i) + td(D, j) - td(i, j)$ for every pair (i, j) of demand points ($td(i, j)$ is the distance between customer i and customer j and D is the depot).

The basic savings concept expresses the savings obtained by joining two routes into one route as illustrated in Figure 4.6.

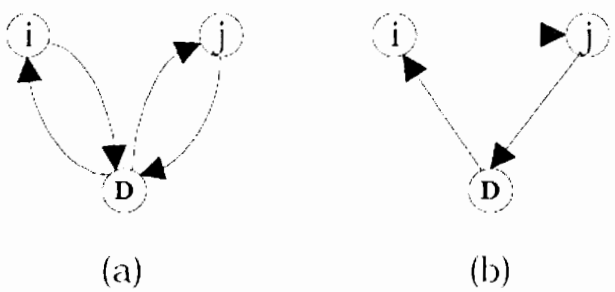

Figure 4.6. Illustration of the savings concept.

Initially in Figure 4.6(a) customers *i* and *j* are visited on separate routes. An alternative to this is to visit the two customers on the same route, as illustrated in Figure 4.6(b). The savings that result from driving the route in Figure 4.6(b) instead of the two routes in Figure 4.6(a) can be calculated. The total travelling distance td_a in Figure 4.6(a) is:

$$td_a = 2*td(D, i) + 2*td(D, j) \quad (5)$$

The total travelling distance td_b in Figure 4.6(b) is:

$$td_b = td(D, i) + td(i,j) + td(D,j) \quad (6)$$

By visiting the two customers on the same route, the savings *s(i,j)*, which is the reduction in travelling distance, is:

$$s(i, j) = td_a - td_b = td(D, i) + td(D, j) - td(i, j) \quad (7)$$

Relatively large values of *s(i,j)* indicate that it is attractive to visit points *i* and *j* on the same route such that point *j* is visited immediately after point *i*.

Step 2. Rank the savings *s(i, j)* and list them in descending order of magnitude. This creates the savings list. Process the savings list beginning with the topmost entry in the list (the largest *s(i, j)*).

Step 3. For the saving *s(i, j)* under consideration, include link *(i, j)* in a route if no route constraints will be violated through the inclusion of *(i, j)* in a route, and if:

- Either, neither *i* nor *j* have already been assigned to a route, in which case a new route is initiated including both *i* and *j*.
- b. Or, exactly one of the two points (*i* or *j*) has already been included in an existing route and that point is not interior to that route (a point is interior to a route if it is not adjacent to the depot in the order of traversal of points), in which case the link *(i, j)* is added to that same route.

- c. Or, both *i* and *j* have already been included in two different existing routes and neither point is interior to its route, in which case the two routes are merged.

Step 4. If the savings list *s(i, j)* has not been exhausted, return to Step 3, processing the next entry in the list; otherwise, stop: the solution to the VRP (Vehicle Routing Problem) consists of the routes created during Step 3. Any points that have not been assigned to a route during Step 3 must each be served by a vehicle route that begins at the depot, visits the unassigned point, and returns to the depot.

The above algorithm does not consider the tour length constraint, and so during coding of the algorithm this constraint is incorporated into the algorithm.

Node Ejection Chains

Ejection Chains are variable depth methods that generate a sequence of interrelated simple moves to create a more complex compound move (Rego & Glover, 2010). There are several types of ejection chains such as node ejection chains and edge ejection chains. As illustrated by Rego & Glover (2010), an ejection chain of L levels consists of a succession of operations performed on a given set of elements, where the kth operation changes the state of one or more elements which are said to be ejected in the $(k+1)th$ operation. This ejection changes the state of other elements, leading in turn to further ejections, until no more operations can be made (according to pre-defined conditions). State-change steps and ejection steps typically alternate, and the options for each depend on the cumulative effect of previous steps (usually, but not necessarily, being influenced by the step immediately preceding). The conditions coordinating the ejection chain process are called legitimacy

conditions, which are guaranteed by associated legitimacy restrictions. The connection between these elements will be clarified subsequently.

A successful node ejection chains algorithm introduced by Rego (2001) is applied to improve the tours and reduce the routing cost. This algorithm includes a multi-node exchange process and a multi-node insert process that are performed sequentially and iteratively in a tabu search heuristic framework. Illustrative diagrams of the multi-node exchange process and multi-node insert process are shown in Figures 4.7 and 4.8 for three levels of an ejection chain.

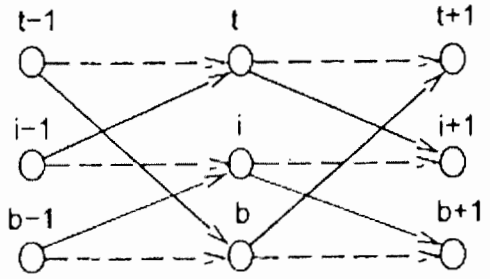

Figure 4.7. Multi-node exchange process (Rego, 2001).

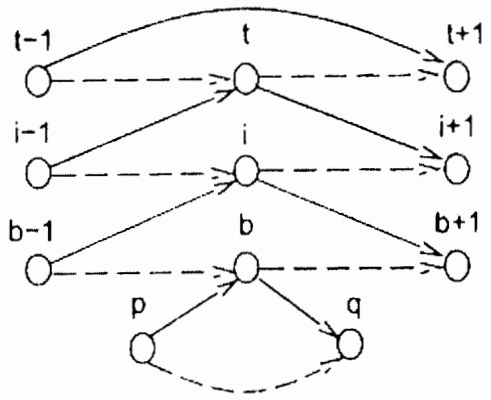

Figure 4.8. Multi-node insert process (Rego, 2001).

As mentioned earlier, the presented LRP is a multi-product LRP and so the customer may demand several types of products with different sizes and volumes. To update the remaining capacity of depots in Procedure 1 and remaining capacity of vehicles in Procedure 2, first a standard volume unit, e.g. one cubic feet, is selected and then the total volume of each customer's demand is calculated based on the total number of standard units for all of the products demanded by the customer. The depot capacity and vehicle capacity are also expressed based on the standard unit.

Computational Results

The procedures of the presented metaheuristic are coded in MATLAB R2010a. Since in the literature there are no benchmark problems for the four-layer LRP, to test the algorithm, a variety of problems are generated and solved. In these problems, the locations of plants, CDs, RDs and customers, demands of customers, and capacity of depots are generated in a random fashion. For a large problem with 2 plants, 15 candidate CDs, 30 candidate RDs, 350 customers, and 5 products, the results are presented in Figures 4.9, 4.10, and 4.11, which are MATLAB outputs. In Figure 4.9, the top-left plot shows the location of facilities and customers in a 500x250 rectangular area. The dark rectangles are plants; the dark triangles are CDs; the white triangles are RDs; and the circles represent customers. The rest of the plots in this figure are related to the best solution. The top-right and mid-left plots display the allocation of customers to facilities and transshipments paths, respectively, after construction phase of the GRASP. The mid-right and bottom-left show the final solutions of allocation of customers to facilities and transshipments paths, respectively, after local search of the GRASP. And the bottom-right plot shows the tours from facilities to customers.

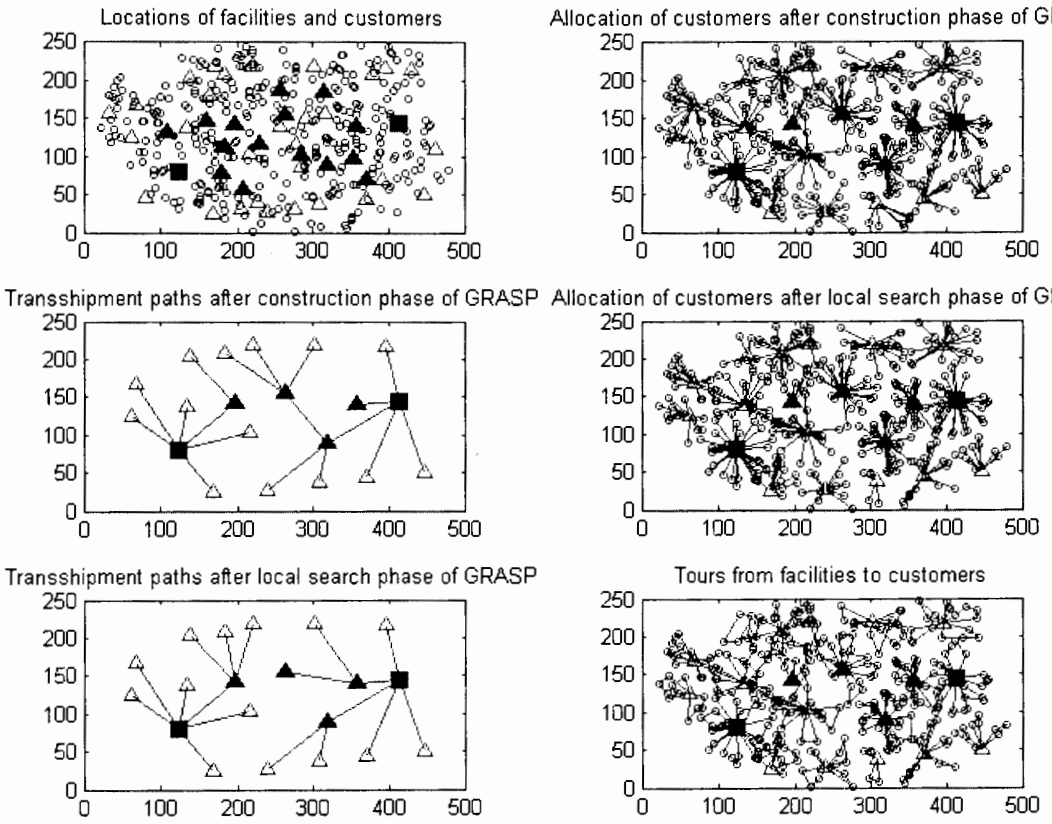

Figure 4.9. Graphical solution.

Figures 4.10 and 4.11 show the results of 2 runs of the algorithm. In each run the total costs of the solutions generated throughout 15 rounds of the GRASP in Procedure 1 are depicted. In each round, 8 iterations of the GRASP were attempted and so a total of 120 solutions were obtained. The elite solutions list size is 5 which means the best 5 solutions obtained so far are kept in the list. As seen, the three strategies of unbiased, intensification, and diversification are applied alternately. Each strategy is executed in 5 rounds. As observed in Figures 4.10 and 4.11, the intensification and diversification strategies are very effective and generate most of the best solutions. They also show that although the

unbaised strategy is the result of the GRASP that generates greedy solutions, the high-quality solutions cannot be achieved in a reasonable amount time unless the intensification and diversification strategies are incorporated into the GRASP. Through numerous runs, it was observed that the best solution is usually obtained during the intensification implementations. The diversification strategy also produces high-quality solutions that can be included in the elite solutions list and be used in the next intensification implementations.

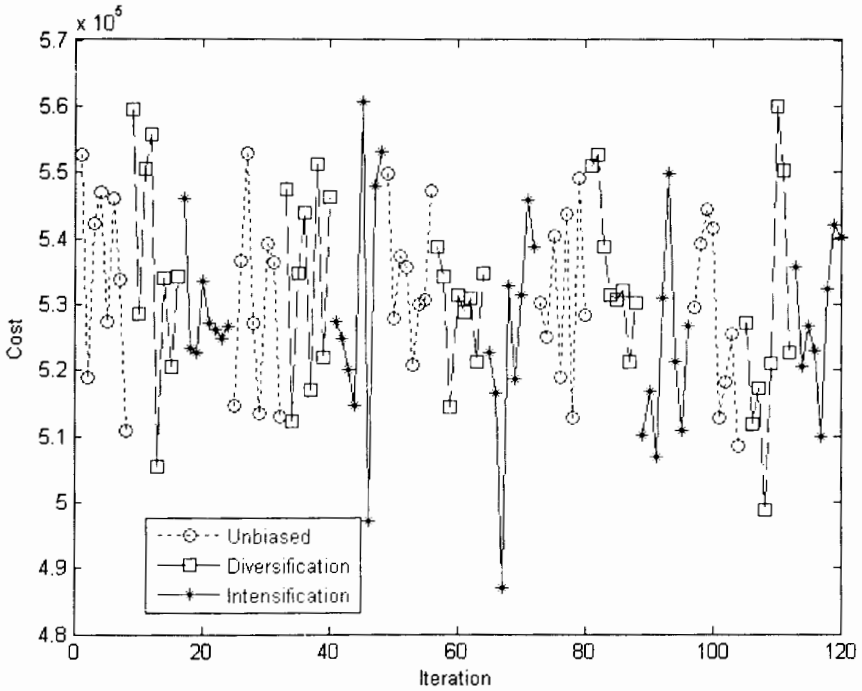

Figure 4.10. Results of unbiased, intensification, and diversification strategies for Scenario 1 (Run 1).

Figure 4.12 shows the effect of the local search procedure of the GRASP in each iteration. As seen, the local search techniques are effective and improve the solutions generated by the construction phase of the GRASP by 2 to 18 percent.

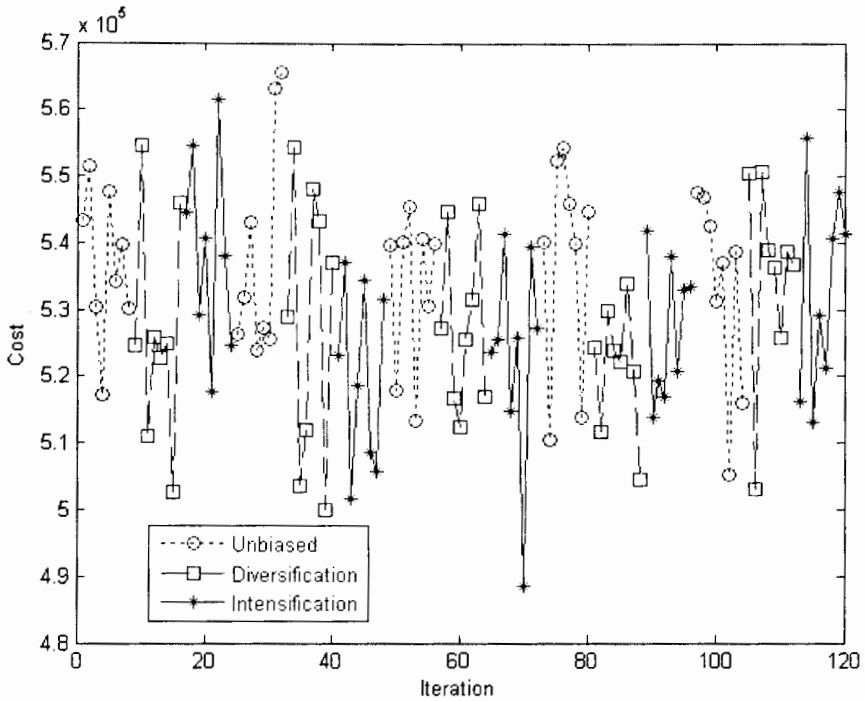

Figure 4.11. Results of unbiased, intensification, and diversification strategies for Scenario 1 (Run 2).

The running time for the above problem on an Intel Core 2 Quad CPU (2.66GHz) machine with 1.97GB of RAM is almost 145 seconds. As reported by Duhamel et al. (2010), the average CPU time for solving large-size two-layer LRPs, instances with 200 customers, are 965 seconds for Prodhon's instances with 10 depots and 1255 seconds for Tuzun and Burke's instances. Also, the CPU times reported by Yu et al. (2010) for solving instances with 200 customers and 10 depots have an average of 1485 seconds. Considering the fact that the presented algorithm in this study solves a four-layer LRP with 2 plants, 15 CDs, 30 RDs, and 350 customers, the CPU time of 145 seconds is a very reasonable amount of time for such a large complex problem.

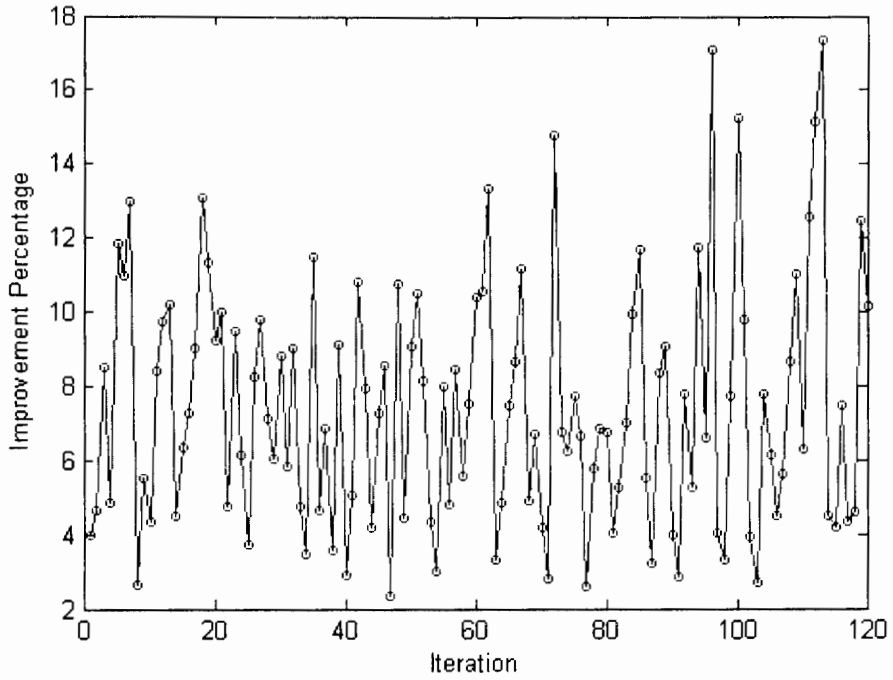

Figure 4.12. Effect of the local search techniques.

To test the algorithm a small problem, similar to the small problem that we solved by the mixed integer programming model in Hamidi et al. (2011) and Hamidi et al. (2012a), was solved. The problem consists of 2 plants, 2 CDs, 2 RDs, and 4 customers. Different versions of the problem with different facility locations were solved, and in all cases the optimal solution was obtained by the algorithm in a very short amount of time, less than 1 second. The solution generated by the algorithm was verified to be optimal by enumeration.

To check the consistency and reliability of the model in different runs, the large problem introduced above, with 2 plants, 15 candidate CDs, 30 candidate RDs, and 350 customers, was solved 30 times. The best solution found in each run is presented in Table 4.1.

Table 4.1. Best solutions of 30 independent runs (Scenario 1).

Run	Best Solution	Run	Best Solution	Run	Best Solution
1	492,631	11	492,767	21	491,291
2	499,355	12	500,973	22	501,595
3	498,781	13	494,847	23	488,890
4	489,020	14	486,990	24	501,703
5	496,560	15	492,349	25	491,842
6	494,520	16	496,413	26	490,427
7	493,077	17	494,889	27	493,997
8	498,001	18	486,934	28	496,478
9	494,450	19	492,506	29	488,428
10	504,380	20	500,576	30	497,201

To show the consistency and reliability of the algorithm, a confidence interval analysis is performed. First a confidence interval for the mean of the best solution generated in each run is calculated. The best solutions in Table 4.1 are generated in 30 independent runs. The sample mean and sample standard deviation of the solutions are: $\overline{X} = 494{,}729$ and $S = 4{,}601$. The sample mean $\overline{X}$ and sample standard deviation S are point estimators of the true population mean μ and the true population standard deviation σ. According to the central limit theorem, the distribution of the sample mean converges to the normal distribution as the sample size gets larger; the sample size should be at least 30. The standard deviation of $\overline{X}$ is $\sigma/\sqrt{n}$. Using the standardized normal distribution:

$$P[\frac{\overline{X} - \mu}{\sigma / \sqrt{n}} \geq -Z_{\alpha}] = 1 - \alpha \qquad (8)$$

where α is the probability that the standardized normal has a value less than $-Z_\alpha$. Using some algebra on Equation (8) gives the following:

$$P[\mu \le \bar{X} + Z_\alpha \frac{\sigma}{\sqrt{n}}] = 1 - \alpha \qquad (9)$$

Equation (9) is the confidence interval for the mean of the best solution. If we choose to calculate a 99% confidence interval (α=0.01), the confidence interval will be:

$$P[\mu \le 494{,}729 + 2.33 \frac{4{,}601}{\sqrt{30}}] = P[\mu \le 496{,}686] = 0.99 \qquad (10)$$

Equation (10) states that with the probability of 99% the mean of the best solution is less than or equal to 496,686. Comparing the upper limit of the confidence interval (496,686) with the 3600 solutions obtained during the 30 runs (120 iterations for each run), it is observed that only 24 solutions out of 3600 solutions (only 0.7% of the solutions) are less than or equal to the upper limit. It should be noted that all of the 3600 solutions are greedy solutions. This shows that the algorithm is really consistent and reliable. This is due to the low standard deviation of the best solutions generated by the algorithm.

By relaxing the limitation of direct shipment distance for shipping products between facilities, the four-layer LRP can be converted to a three-layer LRP (with plants, depots, and customers), in which products can be directly transported to any depot anywhere regardless of the distance. The proposed algorithm can also solve the three-layer LRP; this can be done by setting the maximum direct shipment distance equal to a large number. Figure 4.13 shows the graphical solution of the problem presented above when the direct shipment distance constraint is relaxed.

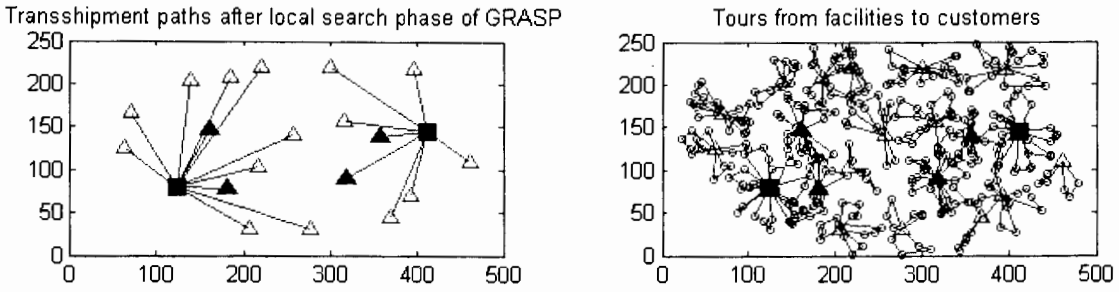

Figure 4.13. Graphical solution of the three-layer LRP.

The above hybrid metaheuristic and computational results are presented and discussed in Hamidi et al. (2012b).

Scenario 2

In the second scenario, the constraints that were relaxed in the first scenario are added to the problem and so all of the constraints are considered. Basically, each plant does not necessarily produce all types of products, and plants have limited production capacity for products. Also, CDs have limited throughput capacity for transshipping products to customers and other depots. As a result, the second scenario is more complex than the first scenario because it considers two more constraints. In this scenario, each plant may transport products to other plants and each depot may receive products from more than one facility. To tackle this scenario, the following solution algorithm is developed.

A Metaheuristic for Scenario 2

The structure of the metaheuristic for Scenario 2 is similar to that of Scenario 1. The schematic structure of the metaheuristic is shown in Figure 4.14. However, the metaheuristic applies a more complex GRASP.

The construction phase and local search phase of the GRASP are different from those of Scenario 1. The construction phase and local search phase are as follows.

Construction Phase of GRASP

In each iteration of the GRASP, the following steps generate a greedy randomized solution. A facility can be a plant, a CD, or an RD. Also, a depot can be a CD or an RD.

Step 1. For each plant, allocate the customers whose distances from the plant are less than or equal to the coverage radius. The coverage radius is the maximum desirable distance of a customer from a facility to which the customer can be allocated. From the practical point of view, it is not desirable to assign distant customers to the facility because multiple customers are to be served in each tour and the tour length is limited. So, this radius can be a fraction (less than ½) of the maximum tour length, e.g. one fourth of the maximum tour length, to make sure that multiple customers are served in one tour.

Step 2. For each plant, calculate the demand of the plant for each product; the demand of the plant is the total demand of the customers assigned to the plant. The minimum of the production capacity of the plant for the product and the demand of the plant is assigned to the plant. If there is unsatisfied demand for the plant, go to Step 3 (the plant is called the facility); otherwise go to Step 8.

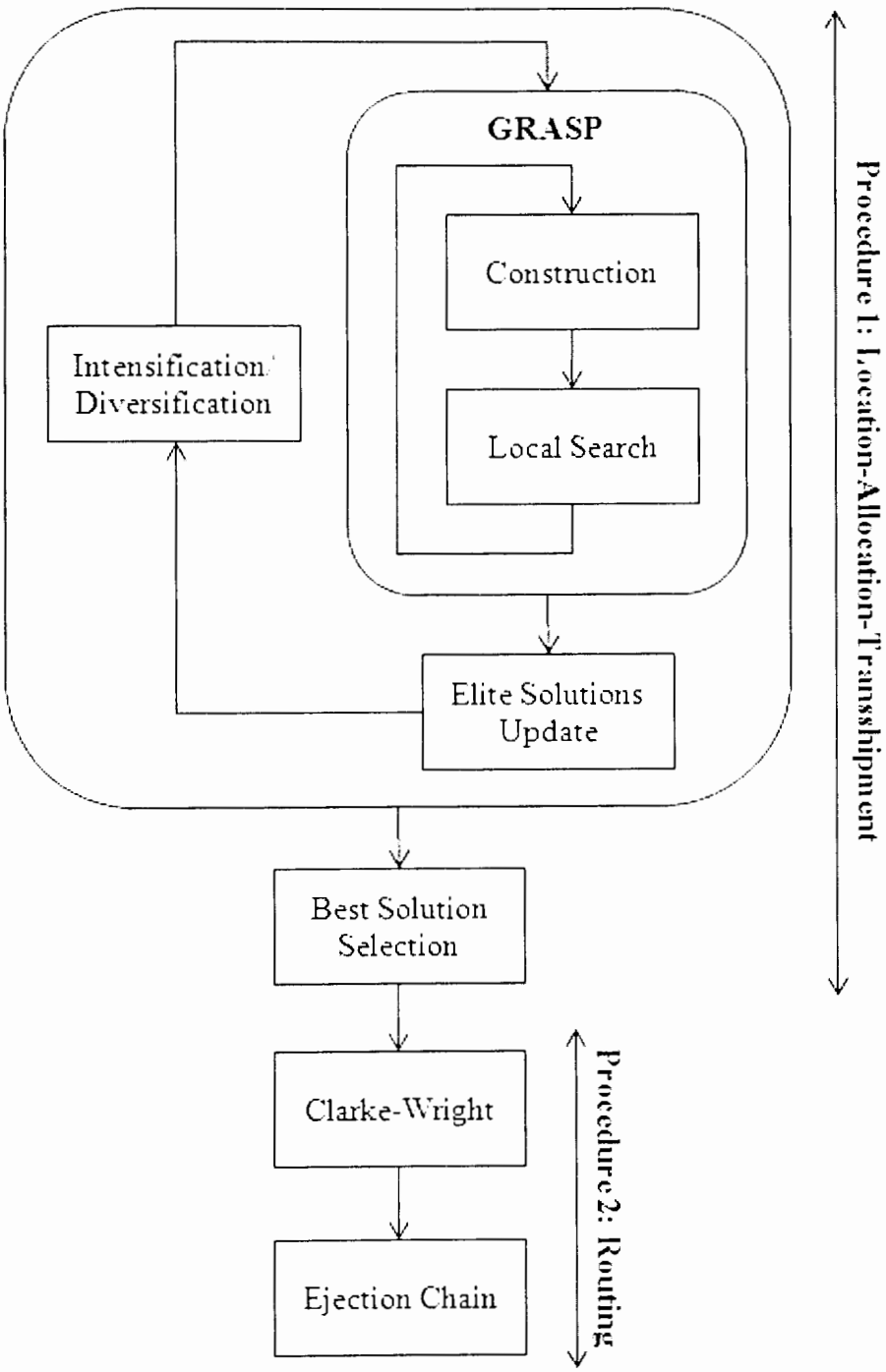

Figure 4.14. Metaheuristic Algorithm.

Step 3. If there is a plant whose distance from the facility is less than the maximum direct shipment distance, the minimum of the remaining production capacity of the plant for the product and the unsatisfied demand of the facility is assigned to the plant, and a transshipment path is constructed from the plant to the facility. If there is unsatisfied demand for the facility, do Step 4 and Step 5 until the demand is satisfied.

Step 4. Select the nearest plant to the facility that has a remaining production capacity for the product more than zero. The minimum of the remaining production capacity of the plant for the product and the unsatisfied demand is called XX.

Step 5. Among plants and open CDs with the following conditions, find the nearest one to the selected plant (called facility B).

- Their distances from the facility are less than the maximum direct shipment distance.
- Their remaining throughput capacity is more than zero.
- Their distances from the selected plant are less than the distance between the facility and the selected plant.

If no such a facility is found, among closed CDs whose distances from the facility are less than the maximum direct shipment distance, find the nearest one to the selected plant (called facility B) and open it. The minimum of XX and the remaining throughput capacity of facility B for the product is called XX1. Construct a transshipment path from facility B to the facility. Increase the unsatisfied demand of facility B by XX1. Reduce XX by XX1. Reduce the unsatisfied demand of the facility by XX1. If XX is not zero, restart Step 5. If XX is zero and there is still unsatisfied demand, go to Step 4.

Step 6. For each opened CD with unsatisfied demand, go to Step 3 to construct the transshipment paths.

Step 7. For each open CD, do the following:

- Assign the near unassigned customers to the depot until the remaining capacity of the depot is less than the customer's demand or the distance between the depot and the customer is more than the coverage radius; sort the unassigned customers based on their distances to the depot and start from the nearest customer.
- For each product, calculate the demand of the facility.
- Go to Step 3 to construct the transshipment paths.

Step 8. Create the main list. Add all of the plants and already opened CDs to the main list; the main list includes all plants and open CDs, some of which will be added to the list in following steps. Select the first facility in the main list.

Step 9. Create the restricted candidate list (RCL) for the selected facility. The RCL for the facility is the list of depots with the following characteristics:

- The depot is unopened.
- The nearest customer to the depot has not been assigned to any facility.
- The depot can directly be connected to the selected facility; its distance to the facility is less than or equal to the maximum direct shipment distance.
- If the selected facility is a CD, only depots that cannot directly be connected to any plant can be in the RCL.

Step 10. From the RCL of the selected facility, randomly select a depot and do the following:

- Open the depot.

- If the depot is a CD, add it to the end of the main list.
- Assign the near unassigned customers to the depot until the remaining capacity of the depot is less than the customer's demand or the distance between the depot and the customer is more than the coverage radius; sort the unassigned customers based on their distances to the depot and start from the nearest customer.
- Do Steps 3 to 7 to construct the transshipment paths.

Step 11. If all customers are assigned, finish the construction phase.

Step 12. Select the next facility in the main list (in case the main list is exhausted, the next facility is the first facility in the main list) and update its RCL. If the RCL is not empty, go to Step 10. Otherwise, restart Step 12. If the RCLs of all facilities in the main list are empty and at least one customer has not been assigned, go to Step 13.

Step 13. For an unassigned customer, find the nearest closed depot and open it. Assign the near unassigned customers to the depot until the remaining capacity of the depot is less than the customer's demand or the distance between the depot and the customer is more than the coverage radius; sort the unassigned customers based on their distances to the depot and start from the nearest customer. To construct transshipment paths for the opened depot, do Steps 3 to 7. Restart this step until all customers are assigned.

In the GRASP presented above, to generate a greedy feasible solution, the four sub-problems described previously (location, allocation, transshipment, and routing problems) are considered simultaneously, and all of the constraints are taken into account. To decrease transshipment and location costs, the depots are attempted to be connected directly or indirectly to the nearest plants; if indirectly, through already opened CDs near to

the plants. To decrease the location cost, the algorithm tries to assign customers to CDs that are already opened for transshipping products to other depots. It also tries to assign as many customers as possible to a depot in order to use the maximum depot capacity and to avoid opening another depot. To decrease the routing cost, the nearest customers are assigned to each depot. Moreover, because the depot cannot be included in the RCL if the nearest customer to the depot has been assigned to any facility, the routing cost and the number of opened depots decrease, which leads to decrease in the location cost.

Local Search of the GRASP

This phase improves the solution developed in the construction phase and finds the local optimal solution. As shown in Figure 4.15, the local search has two sub-procedures which are attempted sequentially and iteratively until no improvement is achieved. The sub-procedures are described below.

```
while "Improvement is achieved"
    Do Transshipment Path Swap;
    Do Depot Drop;
End
```

Figure 4.15. Local Search procedure of the GRASP (Scenario 2).

Transshipment Path Swap: Considering transshipment paths of each product, for every open facility (A) select an open facility (B) and an open facility (C) with the following conditions:

- C sends the product to B, and B sends the product to A.
- The amount of the product sent from C to B is greater than or equal to that of sent from B to A.

For each pair of B and C, if the distance between A and C is less than the maximum direct shipment distance, instead of indirectly transshipping the product from C to A through B, use the direct path from C to A to transport the product directly from C to A. Otherwise, find an open CD or a plant (D) with the following conditions:

- Its distances from A and C are less than the maximum direct shipment distance.
- It has enough remaining capacity for transshipping the current flow of the product from B to A.

If the total distance from C to D and D to A is less than the total distance from C to B and B to A, replace B with D for transshipping the product from C to A.

Depot Drop: For each open RD, if no customer has been assigned to the RD, close it. For each open CD, if no customer has been assigned to the CD and the CD is not a transshipment point, through which products are not transported to other depots, close it.

The above sub-procedures have different impacts on the total cost. "Transshipment Path Swap" reduces the transshipment cost, and "Depot Drop" reduces the location cost.

Diversification and Intensification Using Probabilistic Tabu Search

Like the first scenario, to perform each round of the GRASP, three strategies of unbiased, intensification, and diversification are applied alternately for the second scenario. The intensification strategy generates solutions that are similar to the generated elite solutions by favoring the most frequently selected depots existing in the elite solutions. To perform the intensification strategy, a short-term frequency-based memory is used. This memory consists of the number of times each depot is selected in the current elite solutions. When the intensification strategy is applied, in the construction phase of the GRASP (Step 10), the depots with a higher frequency are selected with a higher probability than the

depots with a lower frequency. In fact, the selection probability is proportional to the frequency.

The diversification strategy generates solutions that are different from generated solutions by favoring the least frequently selected depots. To perform the diversification strategy, a long-term frequency-based memory is used. This memory consists of the number of times each depot is selected in all of the solutions generated so far. When the diversification strategy is applied, in the construction phase of the GRASP (Step 10), the depots with a lower frequency are selected with a higher probability than the depots with a higher frequency. In fact, the selection probability is inversely proportional to the frequency.

The unbiased strategy is the strategy in which none of the intensification and diversification strategies is applied. It is basically a memory-less strategy which gives the procedure the opportunity to produce unbiased and more random solutions.

After m rounds of the GRASP, the best solution is selected. This solution includes location, allocation, and transshipment decisions. Given the solution of Procedure 1, Procedure 2 constructs tours from facilities to customers. Procedure 2 for the second scenario is the same as Procedure 2 described for the first scenario.

Computational Results

The procedures of the presented metaheuristic are coded in MATLAB R2010a. Since in the literature there are no benchmark problems for the four-layer LRP, to test the algorithm, a variety of problems are generated and solved. In these problems, the locations of plants, CDs, RDs and customers, demands of customers, and capacity of depots are generated in a random fashion. For a large problem with 3 plants, 20 candidate CDs, 30

candidate RDs, 380 customers, and 5 products, the results are presented in Figures 4.16 to 4.22, which are MATLAB outputs. Figure 4.16 shows the location of facilities and customers in a 400x400 rectangular area. The dark rectangles are plants; the dark triangles are CDs; the white triangles are RDs; and the circles represent customers. Figure 4.17 is related to the best solution obtained in one run of the algorithm with 120 iterations. In this figure the opened facilities, the allocation of customers to facilities, and the tours from facilities to customers can be viewed.

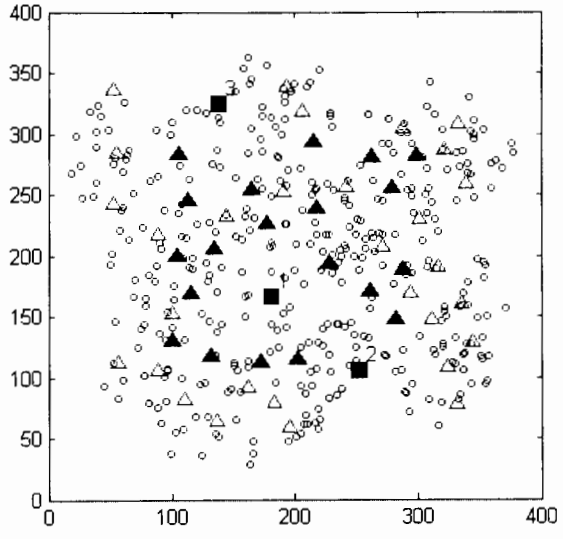

Figure 4.16. Location of facilities and customers.

In this distribution network five products are distributed. Since plants have limited production capacity for products, different scenarios are considered for the production capacity. All three plants have a high production capacity for Product 1. Plant 1 does not produce Product 2, but Plants 2 and 3 have a high production capacity for this product. The only plant producing Product 3 is Plant 1. To produce Product 4, Plants 2 and 3 have a high capacity while Plant 1 has a limited capacity (20% of the total demand of customers). And

to produce Product 5, Plant 1 has a high capacity while Plants 2 and 3 have a limited capacity (20% of the total demand of customers). In Figures 4.18 to 4.22 the transshipment paths for each product before and after the local search phase are shown; the left plot represents the before local search paths, and the right plot represents the after local search paths.

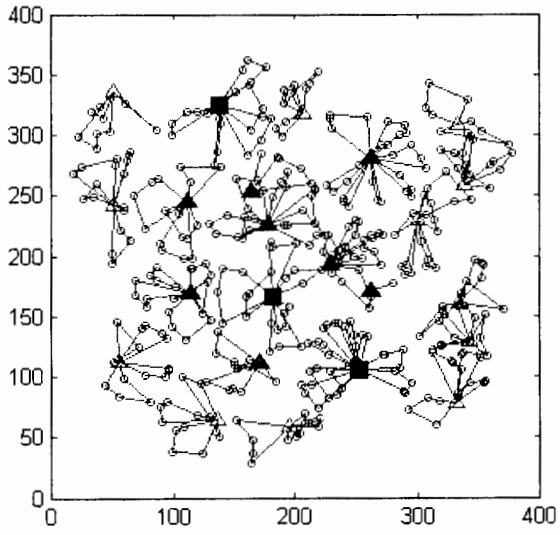

Figure 4.17. Location, allocation, and routing decisions.

Figures 4.23 and 4.24 show the results of 2 runs of the algorithm. In each run the total costs of the solutions generated throughout 15 rounds of the GRASP in Procedure 1 are depicted. In each round, 8 iterations of the GRASP were attempted and so a total of 120 solutions were obtained. The elite solutions list size is 5, which means the best 5 solutions obtained so far are kept in the list. As seen, the three strategies of unbiased, intensification, and diversification are applied alternately. Each strategy is executed in 5 rounds. As observed in Figures 4.23 and 4.24, the intensification strategy is very effective and generates most of the best solutions. The diversification and unbiased strategies also

produce a few good solutions that can be included in the elite solutions list and be used in the next intensification implementations. These strategies help the algorithm to search the solution space more effectively. Basically, by using these strategies the algorithm is not trapped in one or a few attractive region(s) of the solution space that are searched during the intensification strategy, and as a result in a shorter amount of time more regions of the solution space will be searched. Figures 4.23 and 4.24 also show that although the GRASP generates greedy solutions, the high-quality solutions cannot be achieved in a reasonable amount time unless the intensification and diversification strategies are incorporated into the GRASP.

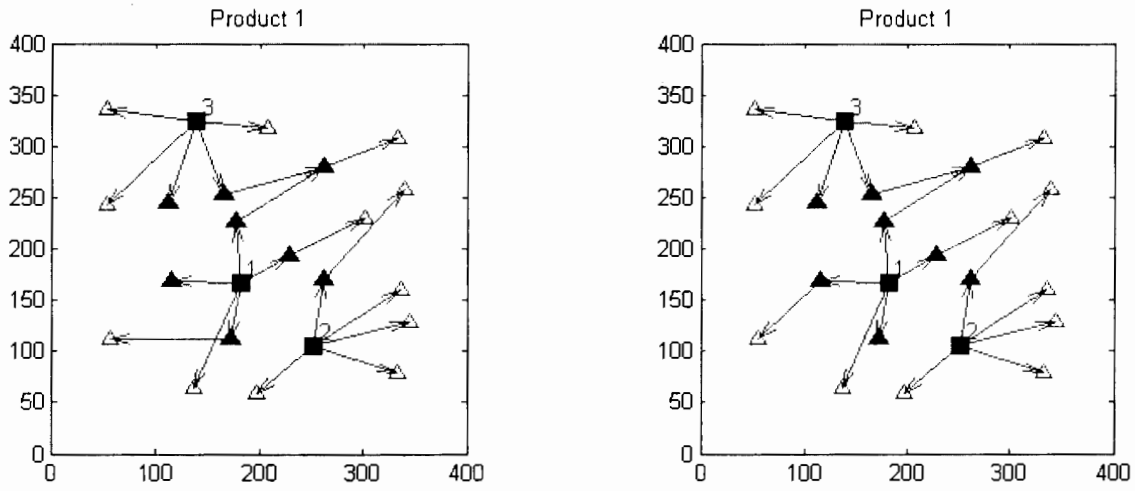

Figure 4.18. Transshipment paths (Product 1).

The running time for the above problem on an Intel(R) Core(TM) i5 CPU (3.19GHz) machine with 3.17 GB of RAM is almost 165 seconds. As reported by Duhamel et al. (2010), the average CPU time for solving large-size two-layer LRPs, instances with 200 customers, are 965 seconds for Prodhon's instances with 10 depots and 1,255 seconds for Tuzun and Burke's instances. Also, the CPU times reported by Yu et al. (2010) for

solving instances with 200 customers and 10 depots have an average of 1,485 seconds. Considering the fact that the presented algorithm in this paper solves a four-layer LRP with 3 plants, 20 CDs, 30 RDs, and 380 customers, the CPU time of 165 seconds is a very reasonable amount of time for such a large complex problem.

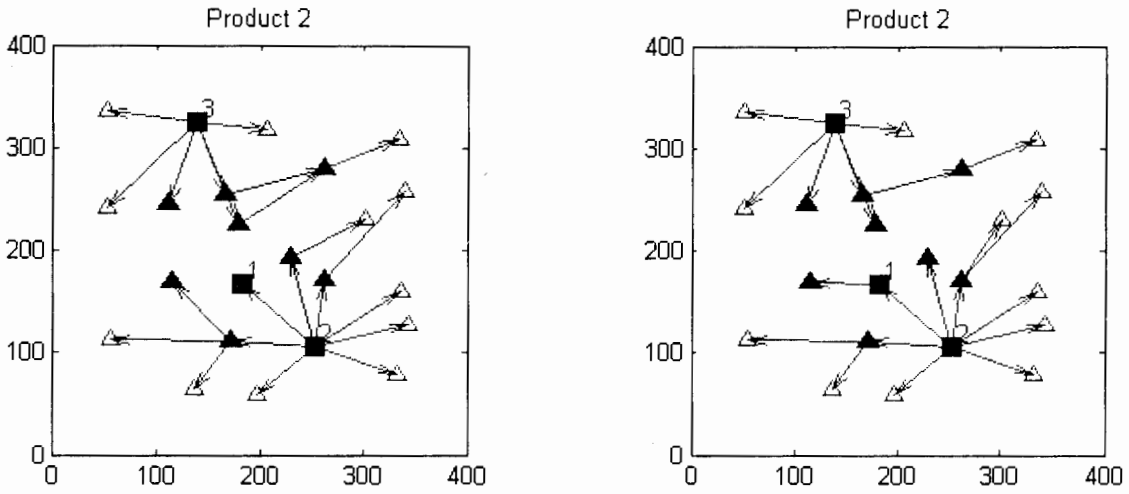

Figure 4.19. Transshipment paths (Product 2).

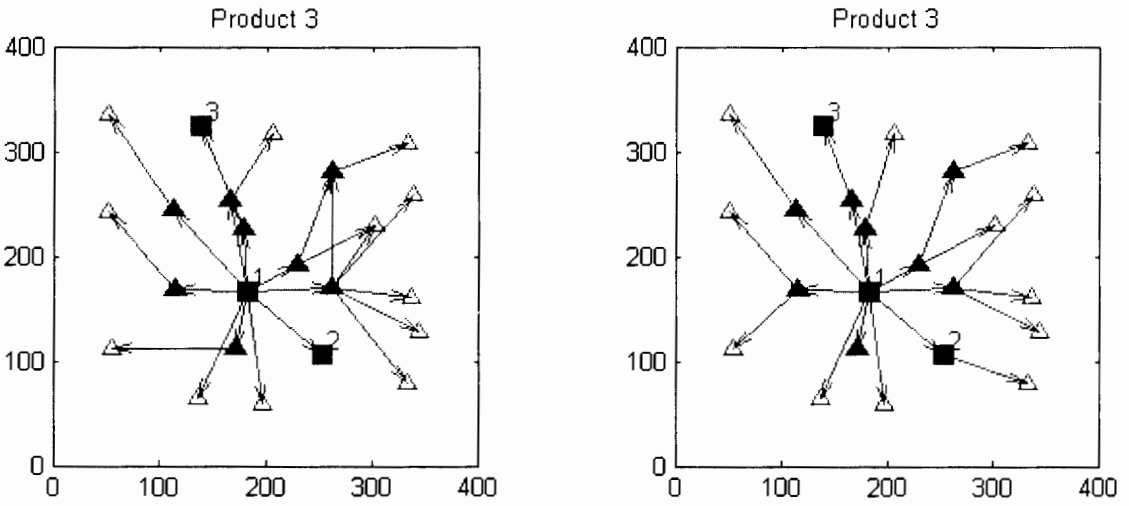

Figure 4.20. Transshipment paths (Product 3).

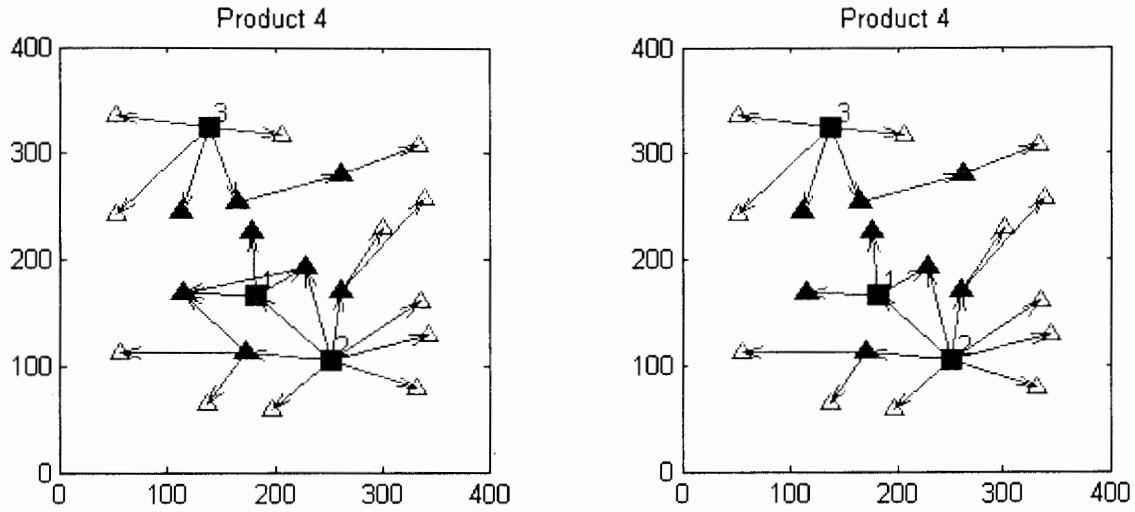

Figure 4.21. Transshipment paths (Product 4).

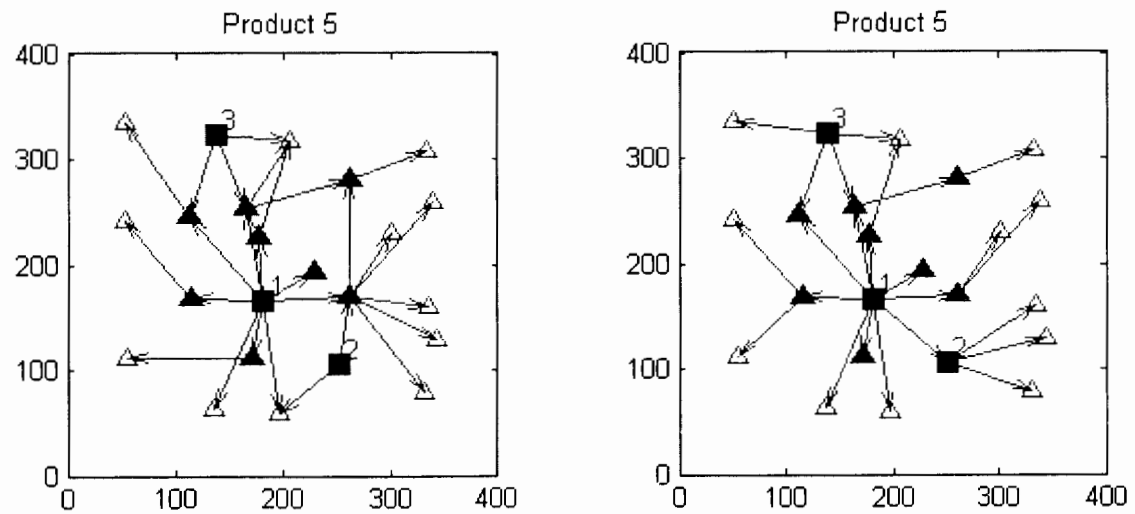

Figure 4.22. Transshipment paths (Product 5).

To test the algorithm, a small problem, similar to the small problem that we solved by the mixed integer programming model in Hamidi et al. (2011) and Hamidi et al. (2012a), was solved. The problem consists of 2 plants, 2 CDs, 2 RDs, and 4 customers.

Different versions of the problem with different production capacities for plants were solved, and in all cases the optimal solution was obtained by the algorithm in a very short amount of time, almost 2 seconds. The solution generated by the algorithm was verified to be optimal by enumeration.

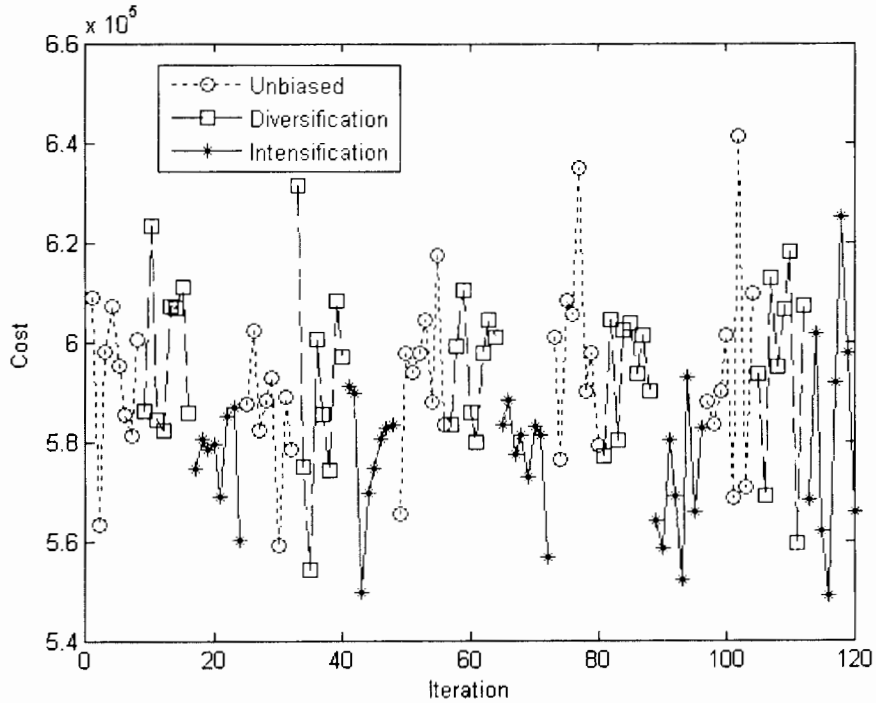

Figure 4.23. Results of unbiased, intensification, and diversification strategies for Scenario 2 (Run 1).

To check the consistency and reliability of the model in different runs, the large problem introduced above, with 3 plants, 20 candidate CDs, 30 candidate RDs, and 380 customers, was solved 30 times. The best solution found in each run is presented in Table 4.2.

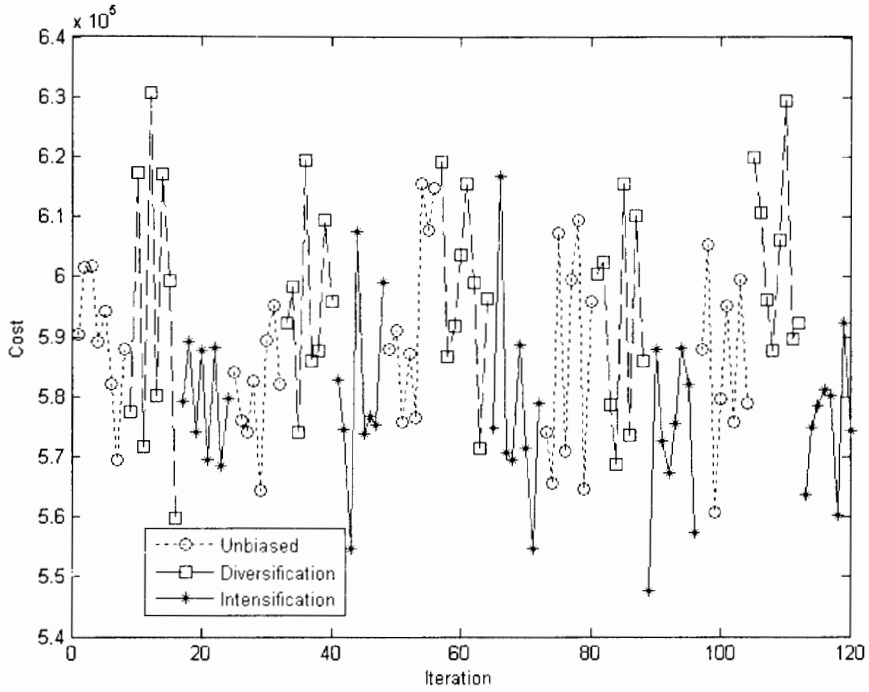

Figure 4.24. Results of unbiased, intensification, and diversification strategies for Scenario 2 (Run 2).

To show the consistency and reliability of the algorithm, a confidence interval analysis is performed. First a confidence interval for the mean of the best solution generated in each run is calculated. The best solutions in Table 4.2 are generated in 30 independent runs. The sample mean and sample standard deviation of the solutions are: $\overline{X} = 550{,}989$ and $S = 5{,}478$. The sample mean $\overline{X}$ and sample standard deviation S are point estimators of the true population mean μ and the true population standard deviation σ. According to the central limit theorem, the distribution of the sample mean converges to the normal distribution as the sample size gets larger; the sample size should be at least 30. The standard deviation of $\overline{X}$ is $\sigma/\sqrt{n}$. Using the standardized normal distribution:

$$P[\frac{\overline{X}-\mu}{\sigma/\sqrt{n}} \geq -Z_{\alpha}] = 1-\alpha \qquad (11)$$

where α is the probability that the standardized normal has a value less than $-Z_{\alpha}$. Using some algebra on Equation (11) gives the following:

$$P[\mu \leq \overline{X} + Z_{\alpha}\frac{\sigma}{\sqrt{n}}] = 1-\alpha \qquad (12)$$

Table 4.2. Best solutions of 30 independent runs (Scenario 2).

Run	Best Solution	Run	Best Solution	Run	Best Solution
1	559,014	11	549,870	21	551,519
2	546,043	12	551,755	22	547,543
3	553,556	13	545,893	23	556,564
4	550,421	14	549,359	24	543,495
5	537,039	15	551,208	25	557,512
6	560,612	16	561,156	26	553,278
7	545,858	17	552,841	27	544,502
8	548,423	18	548,814	28	557,512
9	556,498	19	551,212	29	553,278
10	549,296	20	551,107	30	544,502

Equation (12) is the confidence interval for the mean of the best solution. If we choose to calculate a 99% confidence interval (α=0.01), the confidence interval will be:

$$P[\mu \leq 550{,}989 + 2.33\frac{5{,}478}{\sqrt{30}}] = P[\mu \leq 553{,}320] = 0.99 \qquad (13)$$

Equation (13) states that with the probability of 99% the mean of the best solution is less than or equal to 553,320. Comparing the upper limit of the confidence interval

(553,320) with the 3600 solutions obtained during the 30 runs (120 iterations for each run), it is observed that only 33 solutions out of 3600 solutions (only 0.9% of the solutions) are less than or equal to the upper limit. It should be noted that all of the 3600 solutions are greedy solutions. This shows that the algorithm is really consistent and reliable. This is due to the low standard deviation of the best solutions generated by the algorithm.

By relaxing the limitation of direct shipment distance for shipping products between facilities, the four-layer LRP can be converted to a three-layer LRP (with plants, depots, and customers), in which products can be directly transported to any depot anywhere regardless of the distance. The proposed algorithm can also solve the three-layer LRP; this can be done by setting the maximum direct shipment distance equal to a large number. Figure 4.25 shows the transshipment paths for Product 2 when the three-layer LRP is solved.

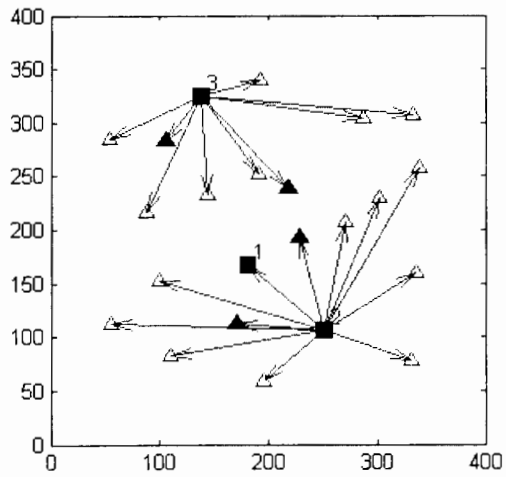

Figure 4.25. Three-layer LRP transshipment paths for Product 2.

The above metaheuristic and computational results are presented and discussed in Hamidi et al. (2012c).

Code Structure

The metaheuristics developed for Scenarios 1 and 2 are coded in MATLAB R2010a. The code consists of a main file and several function files which are called in the main file during the solution process. The code structure is shown is Figure 4.26.

Main File

The structure of the main file is basically the code structure presented in Figure 4.26. First, one of the two scenarios is selected by the user. A module for generating problems with different sizes in a random fashion is incorporated in the main file. The generated problems can be saved and later read. The GRASP module, including the construction phase and local search phase, is run in a loop to generate n solutions in each round of the GRASP. After each round the elite solutions are updated. The GRASP module and elite solution update are attempted in a loop for m rounds. Intensification, diversification, and unbiased strategies are applied alternately during these rounds. After m rounds the best solution is selected among the elite solutions. Then the Clarke-Wright function and ejection chain function are called in the main file to construct tours for the best solution. Also, several modules are coded for depicting the inputs and outputs of the problem. The main file code is presented in Appendix D.

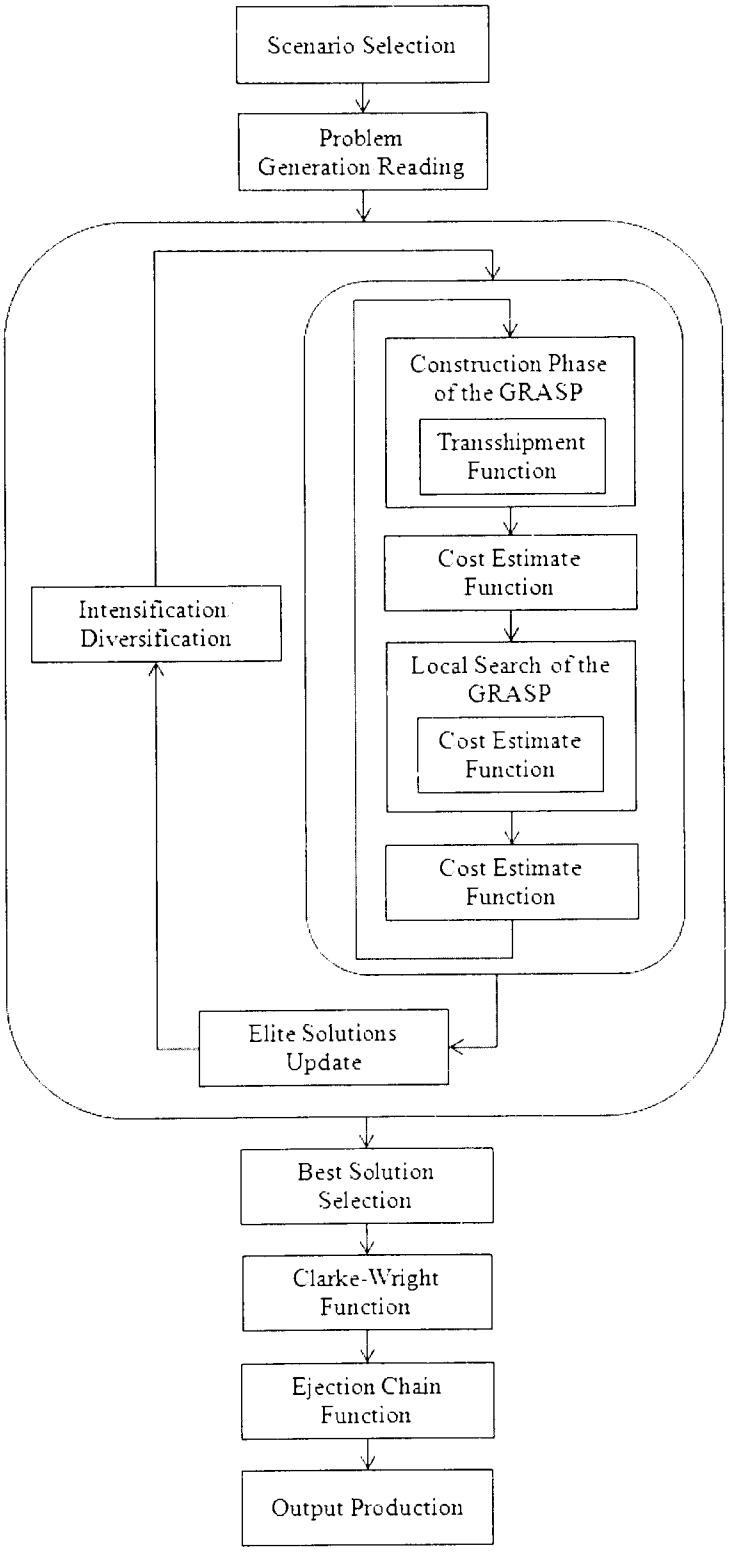

Figure 4.26. Code structure.

Transshipment Function

This function is called frequently during the construction phase of the GRASP in the second scenario. It executes Steps 3 to 5 of the GRASP. The function code is presented in Appendix E.

Cost Estimate Function

This function calculates the estimated total cost for each solution of the GRASP. The estimated total cost includes the location cost, transshipment cost, and estimated routing cost. This function is called after the construction phase of the GRASP, frequently during the local search of the GRASP, and after the local search of the GRASP. The function code is presented in Appendix F.

Clarke-Wright Function

This function executes the Clarke-Wright Savings algorithm. The function code is presented in Appendix G.

Ejection Chain Function

This function runs the node ejection chains algorithm. The function code is presented in Appendix H.

CHAPTER 5. DISCUSSION

The Effects of Problem Size

In Table 5.1 and Figure 5.1 the effect of problem size on computational time (CPU time) is shown. As seen in Table 5.1, eight problem sizes are investigated. The problem size is the sum of the number of plants, the number of CDs, the number of RDs, and the number of customers. For each problem size five different problems are generated in a random fashion and solved by the metaheuristic developed for the first scenario. The CPU time in seconds is the average of CPU times for the five problems. The CPU times for the five problems in each category of problem size are presented in Appendix B. In this analysis an Intel Core 2 Quad CPU (2.66GHz) machine with 1.97GB of RAM is used. Also, for each problem 120 solutions (15 rounds of the GRASP, 8 iterations in each round) are generated.

As seen in Figure 5.1, the CPU time increases with a low slope for problem sizes of 50 to 250. For problems with a size of 250 to 400, the CPU time increases in a linear fashion as the problem size increases. As discussed earlier and seen here, the CPU time is very reasonable even for large-size problems.

In Table 5.2 and Figure 5.2 the superiority of the best solution over the average GRASP solution and worst GRASP solution is shown. In each problem, among all 120 greedy solutions during the GRASP the best solution with the minimum cost and the worst solution with the maximum cost can be realized. The average solution is the average cost of the 120 solutions. The best solution superiority over the average solution is calculated by:

(1 – Best solution cost / Average solution cost). For example, if the superiority is 10%, it means that the best solution is 10% better than the average solution; the best solution cost is 10% less than the average solution cost or in other words the best solution cost is 90% of the average solution cost. Also, the best solution superiority over the worst solution is calculated by: (1 – Best solution cost / Worst solution cost). For each problem size five different problems are generated in a random fashion and solved by the metaheuristic developed for the first scenario. Each superiority is the average of the superiorities for the five problems. The best solution superiorities for the five problems in each category of problem size are presented in Appendix C. As seen in Table 5.2, although the GRASP generates greedy and high-quality solutions, by using probabilistic tabu search (the intensification and diversification strategies) the metaheuristic is able to generate best solutions that are on average 10% better the average greedy solutions and 19% better than worst greedy solutions.

As observed in Figure 5.2, the best solution superiority decreases slightly as the problem size increases. As the problem size increases, the RCL (Restricted Candidate List) of the GRASP becomes more restricted and selective because the RCL for each facility in the main list (each plant or open CD) contains a smaller percentage of facilities. In other words, as the problem size increases, the GRASP becomes greedier and generates better solutions. Therefore, the quality of GRASP solutions increases and consequently the quality of the average solution and worst solution increases. That is why the best solution superiority decreases slightly as the problem size increases.

Table 5.1. Effect of problem size on CPU time.

Number of Plants	Number of CDs	Number of RDs	Number of Customers	Problem Size	CPU Time (Sec.)
2	2	9	37	50	2
2	4	12	82	100	6
2	6	15	127	150	15
2	8	18	172	200	33
2	10	21	217	250	56
2	12	24	262	300	99
2	14	27	307	350	133
2	16	30	352	400	188

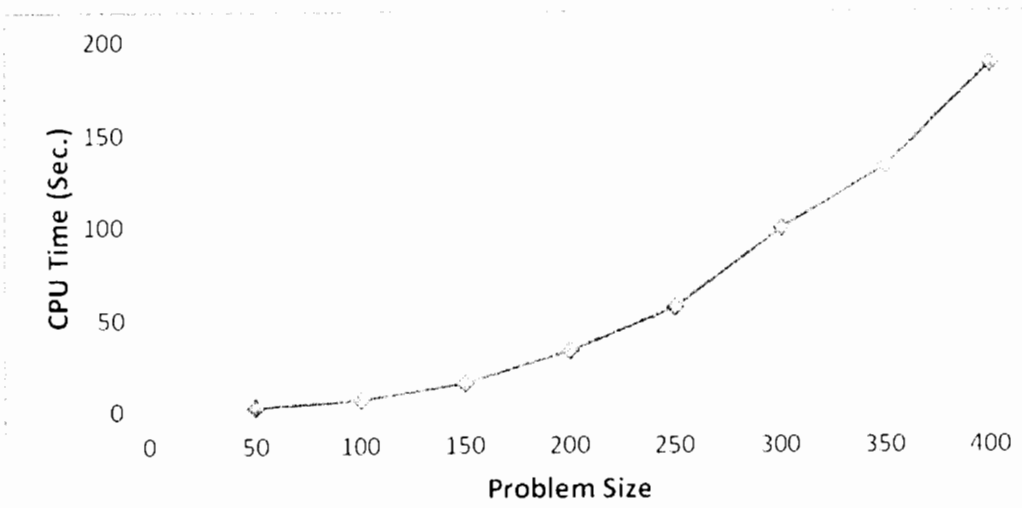

Figure 5.1. Effect of problem size on CPU time.

Table 5.2. Best solution superiority.

Number of Plants	Number of CDs	Number of RDs	Number of Customers	Problem Size	Superiority over Average Solution	Superiority over Worst Solution
2	2	9	37	50	13%	24%
2	4	12	82	100	12%	22%
2	6	15	127	150	10%	19%
2	8	18	172	200	10%	18%
2	10	21	217	250	11%	19%
2	12	24	262	300	9%	15%
2	14	27	307	350	8%	16%
2	16	30	352	400	7%	16%
				Average	10%	19%

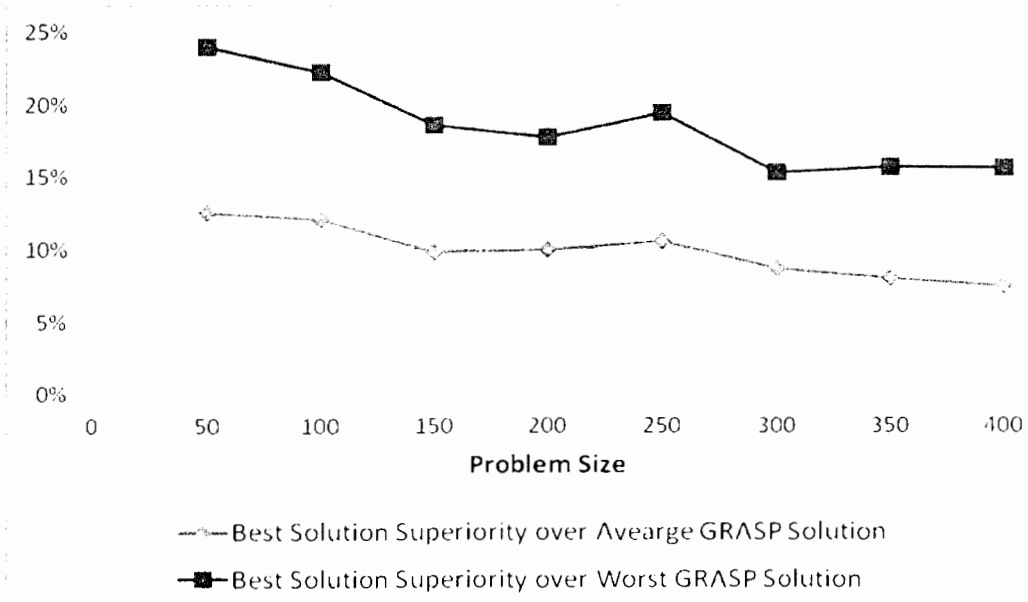

Figure 5.2. Best solution superiority.

Comparison with Other Heuristics

As seen in the literature review, three four-layer LRPs have previously been studied by Bruns et al. (2000), Ambrosino and Scutella (2005), and Lee et al. (2010). By using some assumptions, Bruns et al. (2000) reduced the problem to a two-layer LRP and then to a simple location problem. Basically Bruns et al. (2000) did not solve the four-layer LRP and did not present a solution algorithm for the four-layer LRP. Ambrosino and Scutella (2005) just developed mathematical models for a four-layer LRP and did not propose any solution algorithm that can solve even small-size or mid-size problems.

The only heuristic for a four-layer LRP was developed by Lee et al. (2010). As discussed in Chapter 2, the structure of Lee et al.'s (2010) LRP is different from the four-layer LRP studied in this research. In fact Lee et al.'s (2010) LRP has a different structure compared to usual LRPs as it considers suppliers as one of the layers. In Table 5.3 the CPU times for different-size problems that Lee et al. (2010) solved are shown. The largest problems they solved include 30 suppliers, 10 manufacturers, 10 distribution centers, and 30 customers; since they solved four problems with this size, the average CPU time for the four problems is shown in Table 5.3. As seen in Table 5.3, Lee et al.'s (2010) heuristic needs a considerable amount of time for even small-size problems. Also, the CPU time increases significantly as the problem size increases. This shows that Lee et al.'s (2010) heuristic may not be usable for mid-size or large-size problems.

In Tables 5.4 and 5.5 the average CPU times for different-size problems are presented for Yu et al.'s (2010) and Duhamel et al.'s (2010) heuristics, two recent and high-quality two-layer LRP heuristics. As seen in Tables 5.4 and 5.5, Yu et al.'s (2010) and Duhamel et al.'s (2010) heuristics need a considerable amount of time for even mid-size

problems. Also, for Yu et al.'s (2010) and Duhamel et al.'s (2010) heuristics the CPU time increases significantly as the problem size increases. This shows that these heuristics may not be usable for large-size problems.

Table 5.3. Lee et al.'s (2010) CPU times.

Number of Suppliers	Number of Manufacturers	Number of Distribution Centers	Number of Customers	Problem Size	CPU Time (Sec.)
3	2	2	3	10	1.5
4	2	2	4	12	4.6
5	3	3	5	16	151.1
6	3	3	6	18	256.4
8	3	3	8	22	406.3
30	10	10	30	80	669.2

Table 5.4. Yu et al.'s (2010) CPU times (Tuzun and Burke's instances).

Number of Depots	Number of Customers	Problem Size	CPU Time (Sec.)
10	100	110	316.4
20	100	120	366.3
10	150	160	699.8
20	150	170	834.5
10	200	210	1316
20	200	220	1425

Table 5.5. Duhamel et al.'s (2010) CPU times (Prodhon's instances).

Number of Depots	Number of Customers	Problem Size	CPU Time (Sec.)
5	20	25	0
5	50	55	5.1
5	100	105	152.5
10	100	110	166.7
10	200	210	964.8

Interrelation between Facility Location and Vehicle Routing

As mentioned previously, the LRP combines facility location and vehicle routing. Facility location and vehicle routing are interrelated problems (Nagy & Salhi, 2007). Rand (1976) indicated that "*many practitioners are aware of the danger of sub-optimizing by separating depot location and vehicle routing.*" However, some researchers ignore this interrelation and solve the location problem without paying attention to routing considerations (Nagy & Salhi, 2007). As mentioned by Nagy and Salhi (2007), "*some researchers object to location-routing on the basis of a perceived inconsistency. They point out that location is a strategic, while routing is a tactical problem: routes can be re-calculated and re-drawn frequently (even daily), depot locations are normally for a much longer period. Thus, they claim that it is inappropriate to combine location and routing in the same planning framework due to their different planning horizons.*" Salhi and Rand (1989) and Salhi and Nagy (1999) investigated this criticism by the empirical analyses for

the static and dynamic versions of LRP respectively. Both studies showed that the ignorance of the interrelation between location and routing problems leads to a suboptimal design of the distribution network. Salhi and Rand (1989) evaluated the effect of ignoring routing when locating depots. The study shows that facility location and vehicle routing are interdependent and have to be considered simultaneously. Salhi and Nagy (1999) also found out that combining the location and routing problems decreases costs over a long planning horizon, within which routes are allowed to change. The authors showed that the integrated LRP consistently produces better solutions than the separated location and routing problems.

Dynamic Aspect of Demand

In reality customer demands have dynamic behavior and change over time. Two types of dynamic behaviors can be considered: a) demand increases and b) demand fluctuates (Nagy & Salhi, 2007).

While routing solution can be changed from time to time, depot locations cannot be modified at short notice (Salhi & Nagy, 1999). In fact, location decisions correspond to long-term commitments as they require high investment (Salhi & Nagy, 1999). Changing location decisions, opening and closing depots, intermittently results in a huge cost over time which is not reasonable. So, two different approaches are applied for the demand behaviors mentioned above. For the case of increasing demand, it is more applicable to locate depots for a fairly long period of time, e.g. one year, and change the locations for the next period of time (Nagy & Salhi, 2007). And for the case of fluctuating demand, it is more applicable to locate depots for the whole planning horizon based on average

forecasted demand and change the routing decisions within the planning horizon as demand changes (Nagy & Salhi, 2007; Salhi & Nagy, 1999). With regard to changing routing decisions, the following two strategies can be applied (Berman et al., 1995):

1) Tour planning on a short-period basis: In this strategy based on the real-time information, tours are constructed from the depot to customers.
2) Priori strategy: Through this strategy priori (master) tours are constructed. The customers without any demand at a particular time are skipped on the tour.

The short-period basis strategy is useful to deal with the dynamic aspect of demand, and the priori strategy is useful to deal with the probabilistic aspect of demand.

For the four-layer LRP discussed in this research, to deal with the dynamic, increasing and/or fluctuating, and probabilistic demand the following recommendation is offered. Procedure 1 of the metaheuristic is solved on an annual basis based on the average forecasted demand for the year. Basically, location decisions and transshipment paths are fixed for one year. However, these decisions may change for the following years as the problem is solved based on new forecasted demands. Within each year, to deal with the dynamic and probabilistic aspects of demand, changes in the routing decisions can be made by using a combination of the short-period basis strategy and the priori strategy. For each time period with a certain demand pattern within the year, e.g. one season or quarter of the year, procedure 2 of the metaheuristic can be run based on the average forecasted demand for that particular time period to construct priori tours.

Based on the following discussion it can be seen that the fluctuating demand is less of a concern in the LRP presented in this research. For a single-product LRP the demand fluctuation of the product determines the fluctuation of demand for the whole distribution

network. However, the four-layer LRP discussed in this study represents a multi-product LRP. When multiple products are distributed, the fluctuation of demand is the aggregation of all products' demand fluctuations. And the possibility of having products with different fluctuation behaviors is absolutely probable. In Figure 5.3 the fluctuating demands for four products are shown. The demand for each product has a different dynamic behavior than the others. As seen in this schematic example, although every single product has high seasonal fluctuations in demand, the total demand has low fluctuations due to the fact that in every season some products have low demand and some have high demand. However, in the worst case scenario in which all products have the same fluctuation demand patterns (similar to a single-product LRP), the strategies discussed above can be applied to deal with demand fluctuations.

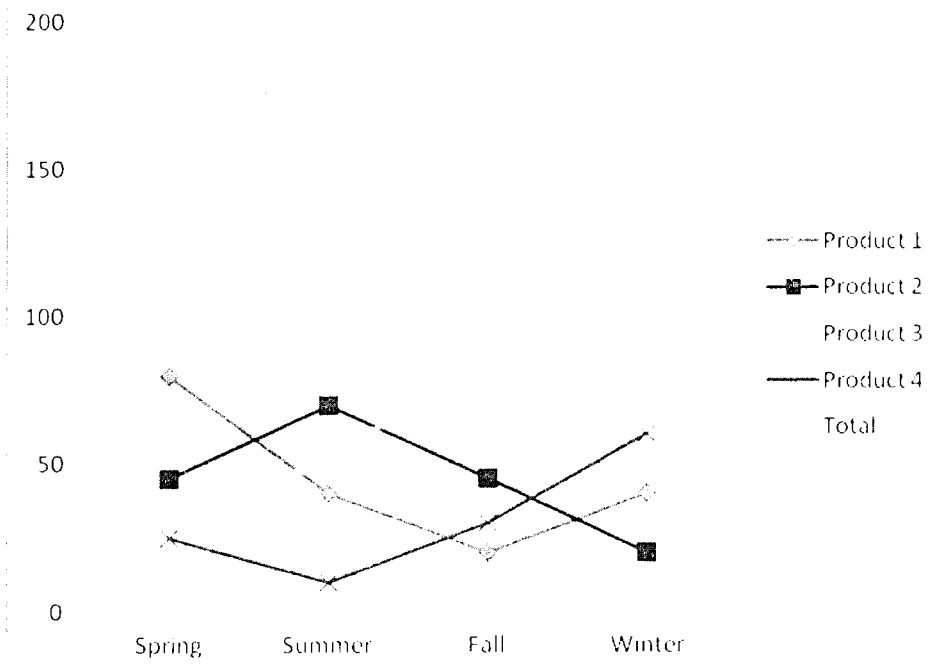

Figure 5.3. Fluctuation of demand.

multi-layer LRPs. The research developed in this study attempts to solve a four-layer LRP that is one the most complex LRPs. As seen in the LRP literature, since LRP is an NP-hard combinatorial optimization problem, in the current state of research, even two-layer LRPs are not optimally solvable and heuristic solutions are still being developed to effectively solve large two-layer LRPs. The fact that the solution algorithms in this study can effectively solve very large and complex four-layer LRPs is a novel and distinct characteristic of this research. As seen in the literature review, the following four-layer LRPs have previously been studied:

- Bruns et al. (2000) studied a four-layer LRP, but as mentioned previously, by using some assumptions they reduced the problem to a two-layer LRP and then to a simple location problem.
- Ambrosino and Scutella (2005) also studied a four-layer LRP. In the study, they just developed mathematical models than can solve only small problems. The authors didn't propose any heuristic to solve even small-size or mid-size problems. Also, Ambrosino and Scutella's (2005) model is a model with one plant and one product that doesn't usually represent real situations. Another drawback of their model is that products cannot be delivered from plants to customers. Also, in their study they didn't consider any limitation for travelling distances.
- Lee et al. (2010) also studied a four-layer LRP. Their LRP has a different structure compared to usual LRPs as they consider suppliers as one of the layers. In their LRP products cannot be delivered from plants to customers. The LRP includes only one layer of depots (distribution centers). In addition, the largest problems they solved include 30 suppliers, 10 potential manufacturers, 10 potential DCs, and 30

customers. The heuristic's computational time for these problems is up to 1,845 seconds. Considering the fact that the problems are not large-size problems, the heuristic needs a considerable amount of computational time for even mid-size problems and may not be applied for large-size problems.

This study also contributes to the knowledge base of combinatorial optimization. Combinatorial optimization is one of the interesting topics in operations research, computer science, and applied mathematics. Applying and combining several metaheuristic and heuristic methods to effectively solve a large and complex combinatorial optimization problem can be viewed as a progress in solving large combinatorial optimization problems. The solution algorithms, as hybrid metaheuristics, can provide solution ideas to researchers who work on complex combinatorial optimization problems in areas of operations research, logistics, and transportation.

CHAPTER 6. CONCLUSION

This dissertation focuses on modeling and solving complicated four-layer and multi-product LRPs, which have not been tackled yet. The LRP integrates location, allocation, vehicle routing, and transshipment problems. Through the modeling phase, the structure, assumptions, and limitations of the distribution network are defined, and the mathematical optimization programming model that can be used to obtain optimal solutions is developed. Since the mathematical model can tackle only small problems, through the solving phase metaheuristic algorithms are developed to solve large-size problems. GRASP, probabilistic tabu search, several local search techniques, the Clarke-Wright Savings algorithm, and a node ejection chains algorithm are combined to solve two versions of the four-layer LRP. Results show that the metaheuristic can solve the problem effectively in terms of computational time and solution quality.

This study contributes to the LRP literature. The fact that the solution algorithm in this study can effectively solve very large and complex four-layer LRPs is a novel and distinct characteristic of this research. By developing solution algorithms that can solve large-size multi-product multi-layer LRPs and produce high-quality solutions in a reasonable amount of time, the main objective of this research has been met.

This study also contributes to the knowledge base of combinatorial optimization. Applying and combining several metaheuristic and heuristic methods to effectively solve a large and complex combinatorial optimization problem can be viewed as a progress in solving these types of problems. The solution algorithms, as hybrid metaheuristics, can

provide solution ideas to researchers who work on complex combinatorial optimization problems in areas of operations research, logistics, and transportation.

LRP models are used to design optimal distribution networks. Distribution is a very important component of logistics and supply chain management. Decreasing the distribution cost leads to decreasing the final cost of the product. Designing new efficient distribution networks and improving existing distribution networks are keys to distribution cost reduction. Integration of location, allocation, routing, and transshipment in the multi-layer LRP helps managers avoid the sub-optimization of distribution solutions. The metaheuristic solution helps companies in designing efficient and economical distribution networks. The presented four-layer LRP, with realistic assumptions and limitations such as producing multiple products, limited plant production capacity, limited depot and vehicle capacity, and limited traveling distances, enables companies to mimic the real world challenges and obtain realistic solutions.

In this study two metaheuristics, GRASP and tabu search, are combined to solve the four-layer LRP. However, the use of other metaheuristics, such as simulated annealing, genetic algorithm, neural networks, and ant colony, can be a venue for future research. Using different combinations of the metaheuristics in hope of solving the four-layer LRP more effectively, in terms of solution quality and computational time, will be an inspiration for researchers in this area.

The four-layer LRP discussed in this study has certain assumptions and limitations which are discussed in Chapter 3. If these assumptions and limitations change, different versions of the four-layer LRP can be obtained and be the subject of future studies. For example, one can assume that direct transportation cannot take place between plants and

regional depots, and plants can send products to central depots only. And/or the maximum direct shipment distance limitation for transportation between plants and central depots is disregarded, which means transportation between plants and central depots can take place regardless of the distance between them. This situation can be considered when plants are in a different country, and sea ports as central depots are the only places that can receive products from plants. Although the four-layer LRP presented in this study is more general and complex than its different versions, formulation and solution of any version of the problem can be the subject of a future work.

REFERENCES

Albareda-Sambola, M., Díaz, J. A., & Fernández, E. (2005). A compact model and tight bounds for a combined location-routing problem. *Computers & Operations Research, 32*(3), 407-428.

Aksen, D., & Altinkemer, K. (2008). A location-routing problem for the conversion to the "click-and-mortar" retailing: The static case. *European Journal of Operational Research, 186*(2), 554-575.

Ambrosino, D., Sciomachen, A., & Scutella, M. (2009). A heuristic based on multi-exchange techniques for a regional fleet assignment location-routing problem. *Computers & Operations Research, 36*(2), 442-460.

Ambrosino, D., & Scutella, M. G. (2005). Distribution network design: New problems and related models. *European Journal of Operational Research, 165*(3), 610-624.

Balakrishnan, A., Ward, J. E., Wong, R. T. (1987). Integrated facility location and vehicle routing models: Recent work and future prospects. *American Journal of Mathematical and Management Sciences, 7*(1-2), 35–61.

Barreto, S. S. (2004). Análise e Modelização de Problemas de localização-distribuição [Analysis and modelling of location-routing problems]. PhD Dissertation, University of Aveiro, Portugal (in Portuguese).

Barreto, S. S., Ferreira, C., Paixão, J., & Santos, B. S. (2007). Using clustering analysis in a capacitated location-routing problem. *European Journal of Operational Research, 179*(3), 968-977.

Berman, O., Jaillet, P., & Simchi-Levi, D. (1995). Location-routing problems with uncertainty. In: Drezner, Z. (Ed.), *Facility Location: A Survey of Applications and Methods*. Springer, New York, 427-452.

Bookbinder, J. H., & Reece, K. E. (1988). Vehicle routing considerations in distribution system design. *European Journal of Operational Research, 37*, 204-213.

Bouhafs, L., Hajjam, A., & Koukam, A. (2006). A combination of simulated annealing and ant colony system for the capacitated location-routing problem. *Lecture Notes in Computer Science, 4251*, 409-416.

Bruns, A., & Klose, A. (1996). A "locate first-route second" heuristic for a combined location-routeing problem. U. Zimmermann, U. Derigs, W. Gaul, R. H. Mohring, K. P. Schuster, eds. *Operations Research Proceedings*. Springer, Braunschweig, Germany.

Bruns, A., Klose, A., & Stahly, P. (2000). Restructuring of Swiss parcel delivery services. *OR Spektrum, 22*, 285-302.

Caballero, R., González, M., Guerrero, F. M., Molina, J., & Paralera, C. (2007). Solving a multiobjective location routing problem with a metaheuristic based on tabu search. Application to a real case in Andalusia. *European Journal of Operational Research, 177*(3), 1751-1763.

Chien, T. W. (1993). Heuristic procedures for practical-sized uncapacitated location-capacitated routing problems. *Decision Sciences, 24*(5), 995-1021.

Christofides, N., & Eilon, S. (1969). Expected distances in distribution problems. *Operational Research Quarterly, 20*(4), 437-443.

Clarke, G., & Wright J. W. (1964). Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research, 12*, 568-581.

Daganzo, C. F. (1984). The distance traveled to visit n points with a maximum of c stops per vehicle: an analytical model and an application. *Transportation Science, 18*(4), 331-350.

Duhamel, C., Lacomme, P., Prins, C., & Prodhon, C. (2008). A memetic approach for the capacitated location routing problem. *Proceedings of the EU/meeting 2008 workshop on metaheuristics for logistics and vehicle routing*. University of Technology of Troyes, France.

Duhamel, C., Lacomme, P., Prins, C., & Prodhon, C. (2010). A GRASP×ELS approach for the capacitated location-routing problem. *Computers & Operations Research, 37*(11), 1912-1923.

Feo, T.A., & Resende, M.G.C. (1995). Greedy randomized adaptive search procedures. *Journal of Global Optimization, 6*, 109-133.

Glover, F., & Laguna, M. (1997). *Tabu Search*. Boston, MA: Kluwer Academic Publishers.

Hamidi, M., Farahmand, K., & Sajjadi, S. R. (2011). Modeling a Four-Layer Location-Routing Problem. *Proceedings of the 2011 IAJC-ASEE International Conference*, Hartford, CT, USA.

Hamidi, M., Farahmand, K., & Sajjadi, S. R. (2012a). Modeling a Four-Layer Location-Routing Problem. *International Journal of Industrial Engineering Computations, 3*(1), 43-52.

Hamidi, M., Farahmand, K., Sajjadi, S. R., & Nygard, K. E. (2012b). A Hybrid GRASP-Tabu Search Metaheuristic for a Four-Layer Location-Routing Problem. *International Journal of Logistics Systems and Management*. Accepted for publication.

Hamidi, M., Farahmand, K., Sajjadi, S. R., & Nygard, K. E. (2012c). A Metaheuristic for a Multi-Product Four-Layer Capacitated Location-Routing Problem. *International Journal of Physical Distribution & Logistics Management.* Under review.

Hansen, P. H., Hegedahl, B., Hjortkjaer, S., & Obel, B. (1994). A heuristic solution to the warehouse location-routing problem. *European Journal of Operational Research, 76*(1), 111-127.

Jacobsen, S. K., & Madsen, O. B. G. (1980). A Comparative Study of Heuristics for a Two-Level Routing-Location Problem. *European Journal of Operational Research, 5*(6), 378-387.

Laporte, G. (1988). Location-routing problems. In: Golden, B. L., & Assad, A. A. (Editors), *Vehicle Routing: Methods and Studies*, North-Holland, Amsterdam. 163-198.

Larsen, R. C., & Odoni, A. R. (2007). *Urban operation research: logistical and transportation planning methods* (2nd edition). Dynamic Ideas Belmont, Massachusetts.

Lee, J., Moon, I., & Park, J. (2010). Multi-level supply chain network design with routing. *International Journal of Production Research, 48*(13), 3957-3976.

Lin, C. K. Y., Chow, C. K., & Chen, A. (2002). A location-routing-loading problem for bill delivery services. *Computers & Industrial Engineering, 43*(1-2), 5-25.

Lin, C. K. Y., & Kwok, R. C. W. (2006). Multi-objective metaheuristics for a location-routing problem with multiple use of vehicles on real data and simulated data. *European Journal of Operational Research, 175*(3), 1833-1849.

Lin, J., & Lei, H. (2009). Distribution systems design with two-level routing considerations. *Annals of Operations Research, 172*(1), 329-347.

Madsen, O. B. G. (1983). Methods for solving combined two level location routing problems of realistic dimensions. *European Journal of Operational Research, 12*, 295-301.

Marinakis, Y., & Marinaki, M. (2008a). A Bilevel Genetic Algorithm for a real life location routing problem. *International Journal of Logistics: Research and Applications, 11*(1), 49-65.

Marinakis, Y., & Marinaki, M. (2008b). A particle swarm optimization algorithm with path relinking for the location routing problem. *Journal of Mathematical modeling and algorithms, 7*(1), 59-78.

May, M.D., & Tu, C.Y. (2008). Location-Routing Based Dynamic Vehicle Routing Problem for Express Pick-Up Service. *IEEE International Conference on Industrial Engineering and Engineering Management.* 1830-1834.

Melechovský, J., Prins, C., & Calvo, R.W. (2005). A Metaheuristic to Solve a Location-Routing Problem with Non-Linear Costs. *Journal of Heuristics, 11*(5-6), 375-391.

Min, H., Jayaraman, V., & Srivastava, R. (1998). Combined location-routing problems: A synthesis and future research directions. *European Journal of Operational Research, 108*(1), 1-15.

Nagy, G, & Salhi, S. (1996a). A nested location routing heuristic using route length estimation. *Studies in Locational Analysis, 10*, 109-127.

Nagy, G, & Salhi, S. (1996b). Nested heuristic methods for the location-routing problem. *The Journal of the Operational Research Society, 47*(9), 1166-1174.

Nagy, G., & Salhi S. (2007). Location-routing: Issues, models and methods. *European Journal of Operational Research, 177*(2), 649-672.

Osman, I. H. & Laporte, G. (1996). Metaheuristics: A bibliography. *Annals of Operations Research 63*. 513- 623.

Perl, J. (1983). A Unified Warehouse Location-Routing Analysis. PhD dissertation, Northwestern University.

Perl, J., & Daskin, M. S. (1985). A warehouse location-routing problem. *Transportation Research B, 19B*(5), 381-396.

Prins, C., Prodhon, C., & Calvo, R.W. (2004). Nouveaux algorithms pour le problème de localisation et routage sous contraintes de capacité. *Proceedings of the MOSIM' 04, Lavoisier, Ecole des Mines de Nantes*, France, 1115-1122.

Prins, C., Prodhon, C., & Calvo, R.W. (2005). A reactive GRASP and Path Relinking algorithm for the Capacitated Location-Routing Problem. *Proceedings of International Conference on Industrial Engineering and Systems Management*, Marrakech, Morocco.

Prins, C., Prodhon, C., & Calvo, R.W. (2006a). A memetic algorithm with population management (MA|PM) for the capacitated location-routing problem. In: *Lecture notes in computer science, 3906*, 183-194.

Prins, C., Prodhon, C., & Calvo, R. W. (2006b). Solving the capacitated location-routing problem by a GRASP complemented by a learning process and a path relinking. *4OR: A Quarterly Journal of Operations Research, 4*(3), 221-238.

Prins, C., Prodhon, C., Ruiz, A., Soriano, P., & Calvo, R. (2007). Solving the Capacitated Location-Routing Problem by a Cooperative Lagrangean Relaxation-Granular Tabu Search Heuristic. *Transportation Science, 41*(4), 470-483.

Prodhon, C. (2007). Solving the capacitated location-routing problem. *4OR: A Quarterly Journal of Operations Research, 5*(4), 339-342.

Rand, G.K. (1976). Methodological choices in depot location studies. *Operational Research Quarterly, 27*, 241-249.

Rego, C. (2001). Node ejection chains for the vehicle routing problem: sequential and parallel algorithms. *Parallel Computing, 27*, 201-222.

Rego, C., & Glover, F.. (2010). Ejection chain and filter-and-fan methods in combinatorial optimization. *Annals of Operations Research, 175*(1), 77-105.

Resende, M. G. C., & Ribeiro, C. C. (2003). Greedy randomized adaptive search procedures. In: Glover, F., & Kochenberger, G., *Handbook of Metaheuristics*, 219-249, Kluwer Academic Publishers.

Resende, M. G. C., & Ribeiro, C. C. (2005). GRASP with path-relinking: recent advances and applications. In: Ibaraki, T., Nonobe, K., & Yagiura, M., *Metaheuristics: Progress as Real Problem Solvers*, Springer, 29-64.

Sajjadi, S. R., Hamidi, M., Farahmand, K., & Khiabani, V. (2011). A greedy heuristic algorithm for the capacitated location-routing problem. *Proceedings of the 2011 Industrial Engineering Research Conference (IERC)*, Reno, NV, USA.

Salhi, S., & Nagy, G. (1999). Consistency and Robustness in Location-Routing. *Studies in Locational Analysis, 13*, 3-19.

Salhi, S., & Rand, G. K. (1989). The effect of ignoring routes when locating depots. *European Journal of Operational Research, 39*(2), 150-156.

Srivastava, R. (1993). Alternate solution procedures for the location-routing problem. *Omega, 21*(4), 497-506.

Stokx, C. F. M., & Tilanus C. B. (1991). Deriving route lengths from radial distances: empirical evidence. *European Journal of Operational Research, 50*(1), 22-26.

The VRP Web. http://neo.lcc.uma.es/radi-aeb/WebVRP/index.html. Retrieved April 24, 2010.

Tuzun, D., & Burke, L. I. (1999). A two-phase tabu search approach to the location routing problem. *European Journal of Operational Research, 116*(1), 87-99.

Wang, X., Sun, X., & Fang, Y. (2005). A two phase hybrid heuristic search approach to the location routing problem. *IEEE international conference on systems, man, and cybernetics, 4*, 3338-3343.

Winston, W. L. (2003). *Operations Research: Applications and Algorithms* (4th edition). Duxbury Press.

Wu, T., Low, C., & Bai, J. (2002). Heuristic solutions to multi-depot location-routing problems. *Computers and Operations Research*, *29*(10), 1393-1415.

Yang, P., & Li-Jun, B. (2006). An integrated optimization problem in logistics and PSO solution. *International Conference on Service System and Service Management 2*, 965-970.

Yu, V. F., Lin, S., Lee, W., & Ting, C. (2010). A simulated annealing heuristic for the capacitated location routing problem. *Computers & Industrial Engineering*, *58*(2), 288-299.

APPENDIX A: GAMS MODEL

```
set
g point /1*10/ alias (g,h,e);

set
f(g) facility /1*6/
p(g) plant /1,2/
pcd(g) plant-CD /1*4/
cd(g) CD /3,4/
rd(g) RD /5,6/
de(g) depot:CD-RD /3*6/
c(g) customer /7*10/
k tour /1*2/
m product /1,2/

s1(g) subset 1 /1,2,3,4,5,6/
ss1(g) subset 1 complement /7,8,9,10/

s2(g) subset 2 /1,2,3,4,5,6,7/
ss2(g) subset 2 complement /8,9,10/

s3(g) subset 3 /1,2,3,4,5,6,8/
ss3(g) subset 3 complement /7,9,10/

s4(g) subset 4 /1,2,3,4,5,6,9/
ss4(g) subset 4 complement /7,8,10/

s5(g) subset 5 /1,2,3,4,5,6,10/
ss5(g) subset 5 complement /7,8,9/

s6(g) subset 6 /1,2,3,4,5,6,7,8/
ss6(g) subset 6 complement /9,10/

s7(g) subset 7 /1,2,3,4,5,6,7,9/
ss7(g) subset 7 complement /8,10/

s8(g) subset 8 /1,2,3,4,5,6,7,10/
ss8(g) subset 8 complement /8,9/

s9(g) subset 9 /1,2,3,4,5,6,8,9/
ss9(g) subset 9 complement /7,10/
```

```
s10(g) subset 10 /1,2,3,4,5,6,8,10/
ss10(g) subset 10 complement /7,9/

s11(g) subset 11 /1,2,3,4,5,6,9,10/
ss11(g) subset 11 complement /7,8/

s12(g) subset 12 /1,2,3,4,5,6,7,8,9/
ss12(g) subset 12 complement /10/

s13(g) subset 13 /1,2,3,4,5,6,7,8,10/
ss13(g) subset 13 complement /9/

s14(g) subset 14 /1,2,3,4,5,6,7,9,10/
ss14(g) subset 14 complement /8/

s15(g) subset 15 /1,2,3,4,5,6,8,9,10/
ss15(g) subset 15 complement /7/;

scalar
tc tour travelling cost /20/
ft tour fixed cost /100/
tl max tour length /200/
vc vehicle capacity /300/
fd max facility distance /150/;

parameter
o(g) depot cost /3 4000
                4 4800
                5 2000
                6 1600/

sc(m) shipment cost /1 0.4
                    2 0.6/

su(m) standard units /1 2
                     2 3/

dc(g) facility space capacity /1 100
                              2 150
                              3 500
                              4 600
                              5 250
                              6 200/;

table td(g,h) distance
```

```
   1   2   3   4   5   6   7   8   9   10
1      230 140 130 180 210 200 180 240 250
2  230     130 150 200 90  240 260 120 110
3  140 130     140 220 90  240 230 120 130
4  130 150 140     100 200 140 150 230 220
5  180 200 220 100     250 50  60  280 260
6  210 90  90  200 250     260 270 50  70
7  200 240 240 140 50  260     70  300 280
8  180 260 230 150 60  270 70      310 290
9  240 120 120 230 280 50  300 310     60
10 250 110 130 220 260 70  280 290 60
```

table d(g,m) demand

	1	2
7	25	10
8	15	5
9	30	15
10	20	10

table pc(g,m) production capacity

	1	2
1	100	50
2	80	60;

```
variables
COST total cost
DepotCost depot cost
TransshipmentCost transshipment cost
TourTravellingCost tour travelling cost
TourFixedCost tour fixed cost;

positive variables
U(g,h,m) shipment units
V(g,m) production units;

binary variable
X(g,h,k) tour routing
Y(g,h) customer allocation
Z(g) depot location;

equations
totcost
depcost
transcost
tourtravcost
tourfixcost
```

```
cons2(g)
cons3(h,k)
cons31(g,k)
cons41
cons42
cons43
cons44
cons45
cons46
cons47
cons48
cons49
cons410
cons411
cons412
cons413
cons414
cons415
cons5(k)
cons6(g,e,k)
cons7(k)
cons8(k)
cons9(h,m)
cons10(g)
cons11(g,m)
cons12(h,m)
cons13(h,m)
cons14(h)
cons15(h)
cons16(g,h,m)
cons01(g,h,m)
cons02(g,m)
cons03(g,h,m)
cons04(g,h,m);

totcost.. COST=e=DepotCost+TransshipmentCost+TourTravellingCost+TourFixedCost;
depcost.. DepotCost=e=sum(dc(g),o(g)*Z(g));
transcost.. TransshipmentCost=e=sum((pcd(g),f(h),m),sc(m)*td(g,h)*U(g,h,m));
tourtravcost.. TourTravellingCost=e=sum((g,h,k),tc*td(g,h)*X(g,h,k));
tourfixcost..  TourFixedCost=e=sum((f(g),c(h),k),ft*X(g,h,k));

cons2(c(g)).. sum((h,k),X(g,h,k))=e=1;
cons3(h,k).. sum(g,X(g,h,k))-sum(g,X(h,g,k))=e=0;
cons31(g,k).. X(g,g,k)=e=0;
cons41.. sum((s1(g),ss1(h),k),X(g,h,k))=g=1;
cons42.. sum((s2(g),ss2(h),k),X(g,h,k))=g=1;
```

```
cons43.. sum((s3(g),ss3(h),k),X(g,h,k))=g=1;
cons44.. sum((s4(g),ss4(h),k),X(g,h,k))=g=1;
cons45.. sum((s5(g),ss5(h),k),X(g,h,k))=g=1;
cons46.. sum((s6(g),ss6(h),k),X(g,h,k))=g=1;
cons47.. sum((s7(g),ss7(h),k),X(g,h,k))=g=1;
cons48.. sum((s8(g),ss8(h),k),X(g,h,k))=g=1;
cons49.. sum((s9(g),ss9(h),k),X(g,h,k))=g=1;
cons410.. sum((s10(g),ss10(h),k),X(g,h,k))=g=1;
cons411.. sum((s11(g),ss11(h),k),X(g,h,k))=g=1;
cons412.. sum((s12(g),ss12(h),k),X(g,h,k))=g=1;
cons413.. sum((s13(g),ss13(h),k),X(g,h,k))=g=1;
cons414.. sum((s14(g),ss14(h),k),X(g,h,k))=g=1;
cons415.. sum((s15(g),ss15(h),k),X(g,h,k))=g=1;
cons5(k).. sum((c(g),f(h)),X(g,h,k))=l=1;
cons6(c(g),f(e),k).. sum(h,X(g,h,k))+sum(h,X(e,h,k))-Y(g,e)=l=1;
cons7(k).. sum((g,h),td(g,h)*X(g,h,k))=l=tl;
cons8(k).. sum((c(g),h,m),d(g,m)*su(m)*X(g,h,k))=l=vc;
cons9(p(h),m).. V(h,m)-sum(f(g),U(h,g,m))-
sum(c(g),d(g,m)*Y(g,h))+sum(f(g),U(g,h,m))=e=0;
cons10(c(g)).. sum(f(h),Y(g,h))=e=1;
cons11(p(g),m).. V(g,m)=l=pc(g,m);
cons12(cd(h),m).. sum(pcd(g),U(g,h,m))-sum(c(g),d(g,m)*Y(g,h))-
sum(de(g),U(h,g,m))=e=0;
cons13(rd(h),m).. sum(f(g),U(g,h,m))-sum(c(g),d(g,m)*Y(g,h))=e=0;
cons14(cd(h)).. sum((c(g),m),d(g,m)*su(m)*Y(g,h))+sum((de(g),m),su(m)*U(h,g,m))-
dc(h)*Z(h)=l=0;
cons15(rd(h)).. sum((c(g),m),d(g,m)*su(m)*Y(g,h))-dc(h)*Z(h)=l=0;
cons16(pcd(g),f(h),m).. td(g,h)*U(g,h,m)-fd*U(g,h,m)=l=0;
cons01(cd(g),p(h),m).. U(g,h,m)=e=0;
cons02(g,m).. U(g,g,m)=e=0;
cons03(c(g),h,m).. U(g,h,m)=e=0;
cons04(rd(g),h,m).. U(g,h,m)=e=0;

model LRP /all/;
solve LRP using mip minimizing COST;
```

APPENDIX B: CPU TIMES

Table B.1. CPU times.

Problem Size	CPU time				
	Problem 1	Problem 2	Problem 3	Problem 4	Problem 5
50	2	2	2	3	2
100	8	6	6	5	6
150	13	17	14	15	17
200	27	35	34	30	38
250	56	43	66	55	59
300	150	94	79	100	71
350	161	127	146	107	122
400	204	203	180	151	201

APPENDIX C: BEST SOLUTION SUPERIORITY

Table C.1. Best solution superiority (problem size 50).

Problem #	Best Solution	Average Solution	Worst Solution	Superiority over Average Solution	Superiority over Worst Solution
Problem 1	77,101	86,113	97,579	10%	21%
Problem 2	80,735	81,972	83,508	2%	3%
Problem 3	86,873	105,056	126,008	17%	31%
Problem 4	74,859	86,531	108,333	13%	31%
Problem 5	71,905	90,048	108,348	20%	34%

Table C.2. Best solution superiority (problem size 100).

Problem #	Best Solution	Average Solution	Worst Solution	Superiority over Average Solution	Superiority over Worst Solution
Problem 1	156,372	174,011	196,561	10%	20%
Problem 2	143,523	164,647	198,426	13%	28%
Problem 3	146,525	164,089	179,153	11%	18%
Problem 4	140,967	165,256	180,836	15%	22%
Problem 5	142,931	162,926	185,013	12%	23%

Table C.3. Best solution superiority (problem size 150).

Problem #	Best Solution	Average Solution	Worst Solution	Superiority over Average Solution	Superiority over Worst Solution
Problem 1	195,011	236,814	270,759	18%	28%
Problem 2	214,208	239,504	260,469	11%	18%
Problem 3	220,285	243,554	272,464	10%	19%
Problem 4	234,211	248,756	275,393	6%	15%
Problem 5	254,885	269,357	293,093	5%	13%

Table C.4. Best solution superiority (problem size 200).

Problem #	Best Solution	Average Solution	Worst Solution	Superiority over Average Solution	Superiority over Worst Solution
Problem 1	291,513	320,766	340,798	9%	14%
Problem 2	306,839	347,350	380,791	12%	19%
Problem 3	309,700	341,258	375,194	9%	17%
Problem 4	290,842	313,018	334,275	7%	13%
Problem 5	263,990	302,793	348,927	13%	24%

Table C.5. Best solution superiority (problem size 250).

Problem #	Best Solution	Average Solution	Worst Solution	Superiority over Average Solution	Superiority over Worst Solution
Problem 1	350,167	392,397	431,538	11%	19%
Problem 2	360,196	396,266	458,033	9%	21%
Problem 3	320,895	367,345	414,717	13%	23%
Problem 4	330,512	379,191	416,422	13%	21%
Problem 5	335,344	362,271	387,674	7%	13%

Table C.6. Best solution superiority (problem size 300).

Problem #	Best Solution	Average Solution	Worst Solution	Superiority over Average Solution	Superiority over Worst Solution
Problem 1	418,063	471,206	525,609	11%	20%
Problem 2	383,549	418,660	452,177	8%	15%
Problem 3	411,895	436,928	460,082	6%	10%
Problem 4	411,081	450,019	484,666	9%	15%
Problem 5	434,415	478,874	513,266	9%	15%

Table C.7. Best solution superiority (problem size 350).

Problem #	Best Solution	Average Solution	Worst Solution	Superiority over Average Solution	Superiority over Worst Solution
Problem 1	449,402	496,838	553,046	10%	19%
Problem 2	482,190	518,469	564,711	7%	15%
Problem 3	447,840	487,999	530,256	8%	16%
Problem 4	468,512	508,119	544,215	8%	14%
Problem 5	469,117	506,405	556,233	7%	16%

Table C.8. Best solution superiority (problem size 400).

Problem #	Best Solution	Average Solution	Worst Solution	Superiority over Average Solution	Superiority over Worst Solution
Problem 1	522,965	561,726	600,377	7%	13%
Problem 2	530,453	571,205	601,304	7%	12%
Problem 3	520,328	565,052	621,821	8%	16%
Problem 4	483,167	517,209	550,836	7%	12%
Problem 5	523,620	573,669	695,620	9%	25%

APPENDIX D: MAIN FILE

```
%     H2_110910

path(path,'Z:\LRP-Matlab\functions');

close all
clear

clear global k X ovc tour EstFixRoutCost EstVarRoutCost EstRoutCost
TransCost DepCost Cost CW ej RE L L2 UU Z opc odc tZ Y DP jj jj1 jj2 jj3
s t u v

global k X ovc tour EstFixRoutCost EstVarRoutCost EstRoutCost TransCost
DepCost Cost CW ej RE L L2 UU Z opc odc tZ Y DP jj jj1 jj2 jj3 s t u v

menu3=menu('Number of Plants','Two Plants','Three Plants');

menu4=menu('Scenario','Scenario 1','Scenario 2');

menu1=menu('Problem','Randomly Generated','Example','Example  2
Layers','Small Example');

if menu1==1

    if menu3==1
        NoPl=2;
    else
        NoPl=3;
    end

    if menu3==1
        NoCD=15;
        NoRD=30;
    else
        NoCD=20;
        NoRD=30;
    end

    NoFac=NoPl+NoCD+NoRD;
    NoCus=380; %350
    vc=75; %
    NoPr=5;
    tc=15;
    ft=100;
    sc=0.3;
    if menu3==1
        fd=120;
    else
```

```
    fd=120;
end
if menu3==1
    tL=120;
else
    tL=150;
end

if menu3==1
    radius=50;
else
    radius=50;
end

A=500;

if menu3==1
    A=500;
else
    A=400;
end

if menu3==1
    B=250;
else
    B=400;
end


if menu3==1
    xx(1)=random('unif',50,A/2-25);
    yy(1)=random('unif',50,B-50);
else
    xx(1)=random('unif',75,A/2);
    yy(1)=random('unif',75,B/2);
end
dc(1)=inf;


if menu3==1
    xx(2)=random('unif',A/2+25,A-50);
    yy(2)=random('unif',50,B-50);
else
    xx(2)=random('unif',A/2,A-75);
    yy(2)=random('unif',75,B/2);
end
dc(2)=inf;

if menu3==2
    xx(3)=random('unif',100,A-100);
    yy(3)=random('unif',B/2,B-75);
end
dc(3)=inf;
```

```
j1=NoPl+1;
while j1<=NoPl+NoCD
    if mod(j1,4)==1
        xx(j1)=random('unif',100,A/2);
        yy(j1)=random('unif',100,B/2);
    end
    if mod(j1,4)==2
        xx(j1)=random('unif',100,A/2);
        yy(j1)=random('unif',B/2,B-100);
    end
    if mod(j1,4)==3
        xx(j1)=random('unif',A/2,A-100);
        yy(j1)=random('unif',100,B/2);
    end
    if mod(j1,4)==0
        xx(j1)=random('unif',A/2,A-100);
        yy(j1)=random('unif',B/2,B-100);
    end

    for j2=1:j1-1
        td(j1,j2)=sqrt((xx(j1)-xx(j2)).^2+(yy(j1)-yy(j2)).^2);
    end
    if (min(td(j1,1:NoPl))>radius)&&(min(td(j1,:))>30)
        dc(j1)=round(random('unif',700,800));
        o(j1)=20*dc(j1);
        j1=j1+1;
    end
end

j1=NoPl+NoCD+1;
while j1<=NoFac
    if mod(j1,4)==1
        xx(j1)=random('unif',50,A/2);
        yy(j1)=random('unif',50,B/2);
    end
    if mod(j1,4)==2
        xx(j1)=random('unif',50,A/2);
        yy(j1)=random('unif',B/2,B-50);
    end
    if mod(j1,4)==3
        xx(j1)=random('unif',A/2,A-50);
        yy(j1)=random('unif',50,B/2);
    end
    if mod(j1,4)==0
        xx(j1)=random('unif',A/2,A-50);
        yy(j1)=random('unif',B/2,B-50);
    end

    for j2=1:j1-1
        td(j1,j2)=sqrt((xx(j1)-xx(j2)).^2+(yy(j1)-yy(j2)).^2);
    end
    if
(min(td(j1,1:NoPl))>radius)&&(min(td(j1,1:NoCD))<=fd)&&(min(td(j1,:))>20)
        dc(j1)=round(random('unif',250,350));
        o(j1)=20*dc(j1);
```

```
            j1=j1+1;
        end
    end

    i=NoFac+1;
    while i<=NoFac+NoCus
        xx(i)=random('unif',0,A);
        yy(i)=random('unif',0,B);
        for j=1:NoFac
            td(i,j)=sqrt((xx(i)-xx(j)).^2+(yy(i)-yy(j)).^2);
        end
        [tdm,j]=min(td(i,:));
        if tdm<=radius
            td(i,j)=inf;
            if min(td(i,:))<=radius
                dp(i,1)=round(random('norm',5,1));
                dp(i,2)=round(random('norm',10,2));
                dp(i,3)=round(random('norm',15,3));
                dp(i,4)=round(random('norm',20,4));
                dp(i,5)=round(random('norm',25,5));

                su(1)=0.5;
                su(2)=0.4;
                su(3)=0.3;
                su(4)=0.2;
                su(5)=0.1;

                d(i)=su*dp(i,:)';
                i=i+1;
            end
        end
    end

    for j1=1:NoFac+NoCus
        for j2=1:NoFac+NoCus
            td(j1,j2)=sqrt((xx(j1)-xx(j2)).^2+(yy(j1)-yy(j2)).^2);
        end
    end
elseif menu1==2
    if menu4==1
        load example
    else
        load example_H2
    end
elseif (menu4==1)&&(menu1==3)
    load example_3layer
    fd=1000;
elseif (menu4==2)&&(menu1==3)
    load example_H2_3layer
    fd=1000;
else
    NoPl=2;
    NoCD=2;
    NoRD=2; %
    NoFac=NoPl+NoCD+NoRD;
```

```
NoCus=4; %
vc=300; %
if menu4==2
    NoPr=2;
    pc(1,1)=30;
    pc(1,2)=50;
    pc(2,1)=80;
    pc(2,2)=20;
end
tc=20;
ft=100;
sc=0.2;
fd=150;
tL=200;
radius=80;

A=400;
B=400;

xx(1)=100;
yy(1)=250;

xx(2)=300;
yy(2)=200;

xx(3)=215;
yy(3)=320;

xx(4)=165;
yy(4)=150;

xx(5)=70;
yy(5)=80;

xx(6)=300;
yy(6)=300;

xx(7)=65;
yy(7)=45;

xx(8)=30;
yy(8)=60;

xx(9)=330;
yy(9)=310;

xx(10)=350;
yy(10)=290;

dc(1)=inf;
dc(2)=inf;
dc(3)=500;
dc(4)=600;
dc(5)=250;
```

```
    dc(6)=200;

    o(3)=4000;
    o(4)=4800;
    o(5)=2000;
    o(6)=1600;

    dp(7,1)=25;
    dp(7,2)=10;
    dp(8,1)=15;
    dp(8,2)=5;
    dp(9,1)=30;
    dp(9,2)=15;
    dp(10,1)=20;
    dp(10,2)=10;

    su(1)=2;
    su(2)=3;

    for i=7:10
        d(i)=su*dp(i,:)';
    end

    for j1=1:NoFac+NoCus
        for j2=1:NoFac+NoCus
            td(j1,j2)=sqrt((xx(j1)-xx(j2)).^2+(yy(j1)-yy(j2)).^2);
        end
    end
end

if (menu4==2)&&(menu1~=4)
    if menu3==1
        pc(1,1)=sum(dp(:,1));
        pc(2,1)=sum(dp(:,1));

        pc(1,2)=0*sum(dp(:,2));
        pc(2,2)=1.2*sum(dp(:,2));

        pc(1,3)=1.2*sum(dp(:,3));
        pc(2,3)=0*sum(dp(:,3));

        pc(1,4)=.2*sum(dp(:,4));
        pc(2,4)=1*sum(dp(:,4));

        pc(1,5)=1*sum(dp(:,5));
        pc(2,5)=.2*sum(dp(:,5));
    else
        pc(1,1)=sum(dp(:,1));
        pc(2,1)=sum(dp(:,1));
        pc(3,1)=sum(dp(:,1));

        pc(1,2)=0*sum(dp(:,2));
        pc(2,2)=sum(dp(:,2));
        pc(3,2)=sum(dp(:,2));
```

```
        pc(1,3)=1.2*sum(dp(:,3));
        pc(2,3)=0*sum(dp(:,3));
        pc(3,3)=0*sum(dp(:,3));

        pc(1,4)=.2*sum(dp(:,4));
        pc(2,4)=1*sum(dp(:,4));
        pc(3,4)=1*sum(dp(:,4));

        pc(1,5)=1*sum(dp(:,5));
        pc(2,5)=.2*sum(dp(:,5));
        pc(3,5)=.2*sum(dp(:,5));
    end
end

%----------
figure
for j=1:NoPl
    plot(xx(j),yy(j),'s','MarkerFaceColor','r','MarkerSize',9)
    text(xx(j),yy(j)+5,j,int2str(j),'FontSize',8);
    hold on
end
axis equal
axis([0 A 0 B])
for j=NoPl+1:NoPl+NoCD
    plot(xx(j),yy(j),'^','MarkerFaceColor','r','MarkerSize',8)
    text(xx(j),yy(j)+5,j,int2str(j),'FontSize',8);
    hold on
end
for j=NoPl+NoCD+1:NoFac
    plot(xx(j),yy(j),'^','MarkerSize',6)
    text(xx(j),yy(j)+5,j,int2str(j),'FontSize',8);
    hold on
end


%   -GRASP-----------
tic
ee=1;
EliteSize=5;
rounds=15;
iter=8;
Elite=cell(12,EliteSize);
Overall=cell(5,200);
freq1(1:NoFac)=0;


nn=0;

for rr=1:rounds

    fprintf('\nRound: %g : ',rr);

    for ii=1:iter
```

```
        fprintf('Iteration: %g\n',ii);

        %--- Construction Phase of TSAS
        ----------

        Y(1:NoFac+NoCus,1:NoFac+NoCus)=0;
        Z(1:NoFac)=0;
        odc(1:NoFac)=0;
        if menu4==2
            opc(1:NoPl,1:NoPr)=0;
            UU(1:NoFac,1:NoFac,1:NoPr)=0;
        end

        %----------Plants

        Z(1:NoPl)=1;
        td2(:,:)=td(:,:);
        td2(1:NoFac,:)=inf;
        td2(:,NoFac+1:NoFac+NoCus)=inf;
        for j=NoPl+1:NoFac
            if min(min(td(j,1:NoPl+NoCD)))>fd
                td2(:,j)=inf;
            end
        end

         for i=NoFac+1:NoFac+NoCus
             if min(td(i,1:NoPl))<=radius
                 [~,j]=min(td(i,1:NoPl));
                 Y(i,j)=1;
                 odc(j)=odc(j)+d(i);
                 td2(i,:)=inf;
             end
         end

         if menu4==2
             for jj=1:NoPl
                 %

                 %---
                 tZ(jj)=1;

Transship(NoFac,NoPl,NoCD,td,su,dp,NoPr,pc,fd,dc,radius,d);

             end
             for m=1:NoPr
                 if sum(opc(:,m))~=sum(Y,2)'*dp(:,m)
                     fprintf('
                     error('
                 end
             end

             if nn==0
                 figure
                 subplot(3,2,1);
```

```
plot(xx(1:NoPl),yy(1:NoPl),'s','MarkerFaceColor','k','MarkerSize',9)
                    axis equal
                    axis([0 A 0 B])
                    hold on

plot(xx(NoPl+1:NoPl+NoCD),yy(NoPl+1:NoPl+NoCD),'^','MarkerFaceColor','k',
'MarkerSize',8)
                    hold on

plot(xx(NoPl+NoCD+1:NoFac),yy(NoPl+NoCD+1:NoFac),'^','MarkerSize',6)
                    hold on

plot(xx(NoFac+1:NoFac+NoCus),yy(NoFac+1:NoFac+NoCus),'o','MarkerSize',2)
                    hold off

                    subplot(3,2,2);

plot(xx(1:NoPl),yy(1:NoPl),'s','MarkerFaceColor','k','MarkerSize',9)
                    axis equal
                    axis([0 A 0 B])
                    hold on
                    for j=NoPl+1:NoPl+NoCD
                        if Z(j)==1

plot(xx(j),yy(j),'^','MarkerFaceColor','k','MarkerSize',8)
                        end
                        hold on
                    end
                    for j=NoPl+NoCD+1:NoFac
                        if Z(j)==1
                            plot(xx(j),yy(j),'^','MarkerSize',6)
                        end
                        hold on
                    end
                    for i=NoFac+1:NoFac+NoCus
                        if sum(Y(i,:))==1
                            plot(xx(i),yy(i),'o','MarkerSize',2)
                        else

plot(xx(i),yy(i),'o','MarkerFaceColor','r','MarkerSize',2)
                        end
                        hold on
                    end
                    for i=NoFac+1:NoFac+NoCus
                        for j=1:NoFac
                            if Y(i,j)==1
                                plot(xx([i,j]),yy([i,j]))
                                hold on
                            end
                        end
                    end
                    hold off

                    figure
                    for m=1:NoPr
```

```
                    subplot(3,2,m);
                    for j=1:NoPl

plot(xx(j),yy(j),'s','MarkerFaceColor','c','MarkerSize',9)
                        text(xx(j)-
5,yy(j)+15,j,int2str(j),'FontSize',8);
                        text(xx(j)-10,yy(j)-
15,sum(UU(j,:,m)),int2str(sum(UU(j,:,m))),'FontSize',8);
                        hold on
                    end

                    axis equal
                    axis([0 A 0 B])
                    title(['Product ',num2str(m)]);
                    hold on
                    for j=NoPl+1:NoPl+NoCD
                        if Z(j)==1

plot(xx(j),yy(j),'^','MarkerFaceColor','k','MarkerSize',8)
                            text(xx(j)-
5,yy(j)+15,j,int2str(j),'FontSize',8);
                            text(xx(j)-10,yy(j)-
15,sum(UU(j,:,m)),int2str(sum(UU(j,:,m))),'FontSize',8);

                        end
                        hold on
                    end
                    for j=NoPl+NoCD+1:NoFac
                        if Z(j)==1
                            plot(xx(j),yy(j),'^','MarkerSize',6)

text(xx(j)+5,yy(j)+5,j,int2str(j),'FontSize',8);
                        end
                        hold on
                    end
                    for j1=1:NoPl+NoCD
                        for j2=1:NoFac
                            if sum(UU(j1,j2,m))>0
                                plot(xx([j1,j2]),yy([j1,j2]));
                                hold on
                            end
                        end
                    end
                end
                hold off
            end


        end


        % ... (Can and MT) ... ---- ---- ---


        list(1:NoPl)=1:NoPl;
        if menu4==2
```

```
        for j=NoPl+1:NoPl+NoCD
            if Z(j)==1
                list(1+length(list))=j;
            end
        end
    end
    tabu(NoPl+1:NoFac)=0;

    j4=1;
    nearDep(1:NoPl+NoCD,1:NoPl+NoCD)=1;

    if menu4==1
        U(1:NoFac+NoCus,1:NoFac+NoCus)=0;
        U1(1:NoFac,1:NoFac)=0;
    end

    temp=1;

    while sum(sum(nearDep))>0

        i=NoFac+1:NoFac+NoCus;
        for j=NoPl+1:NoFac
            if ((tabu(j)==0)&&(Z(j)==0))
                [mintd,i1]=min(td(j,i));
                i1=i1+NoFac;
                if sum(Y(i1,:))==1
                    tabu(j)=1;
                end
            end
        end

        nearDep(1:NoPl+NoCD,:)=0;
        nearbySize(1:NoPl+NoCD)=0;

        for j1=1:NoPl
            j3=0;
            for j2=NoPl+1:NoFac
                if ((td(j1,j2)<=fd)&&(tabu(j2)==0)&&(Z(j2)==0))
                    j3=j3+1;
                    nearDep(j1,j3)=j2;
                end
            end
            nearbySize(j1)=j3;
        end

        for j1=NoPl+1:NoPl+NoCD
            if Z(j1)==1
                j3=0;
                for j2=NoPl+1:NoFac
                    if
((j1~=j2)&&(min(td(j2,1:NoPl))>fd)&&(td(j1,j2)<=fd)&&(tabu(j2)==0)&&(Z(j2
)==0))
                        j3=j3+1;
                        nearDep(j1,j3)=j2;
```

```
            end
        end
        nearbySize(j1)=j3;
    end
end

if nearbySize(list(j4))>0
    if mod(rr,3)==1
        Strategy='Unbiased';
        j1=randi(nearbySize(list(j4)));
        j2=nearDep(list(j4),j1);
    elseif mod(rr,3)==0
        Strategy='Intensification';
        freq(1:NoFac)=0;
        for n1=1:EliteSize
            Z3=Elite{4,n1};
            for j=1:NoFac
                if Z3(j)==1
                    freq(j)=freq(j)+1;
                end
            end
        end
        for j=1:NoFac
            if freq(j)==0
                freq(j)=1;
            end
        end

        rn1=0;
        for j8=1:nearbySize(list(j4))
            rn1=rn1+freq(nearDep(list(j4),j8));
        end

        rn=randi(rn1);
        rn1=0;
        for j8=1:nearbySize(list(j4))
            rn1=rn1+freq(nearDep(list(j4),j8));
            if rn<=rn1
                j2=nearDep(list(j4),j8);
                break
            end
        end
    else
        Strategy='Diversification';
        freq2=freq1;
        for j=1:NoFac
            if freq2(j)==0
                freq2(j)=1;
            end
        end

        for j=1:NoFac
            freq2(j)=round(100/freq2(j));
        end
        rn1=0;
```

```
    for j8=1:nearbySize(list(j4))
        rn1=rn1+freq2(nearDep(list(j4),j8));
    end

    rn=randi(rn1);
    rn1=0;
    for j8=1:nearbySize(list(j4))
        rn1=rn1+freq2(nearDep(list(j4),j8));
        if rn<=rn1
            j2=nearDep(list(j4),j8);
            break
        end
    end
end

Z(j2)=1;

i=1:NoFac+NoCus;
td1(i)=td(j2,i);
td1(1:NoFac)=inf;

while min(td1(i))<=radius
    [mintd1,i1]=min(td1(i));
    if sum(Y(i1,:))==0
        if d(i1)<=(dc(j2)-odc(j2))
            Y(i1,j2)=1;
            td2(i1,:)=inf;
            td2(:,j2)=inf;
            odc(j2)=odc(j2)+d(i1);
        else
            break
        end
    end
    td1(i1)=inf;
end

if menu4==1
    U1(list(j4),j2)=1;
    U(list(j4),j2)=U(list(j4),j2)+d*Y(:,j2);
    if list(j4)>NoPl
        temp=1;
        j5=list(j4);
        while temp==1
            [j6,~]=find(U1(:,j5)==1);
            U(j6,j5)=U(j6,j5)+d*Y(:,j2);
            if j6<=NoPl
                break
            else
                j5=j6;
            end
        end
    end
else
    jj=j2;
    tZ(jj)=1;
```

```
                    %----------
                    fprintf('---- ---------- ----------\n');
                    fprintf('\njj: %g\n',jj);
                    fprintf('---------- ---------- ----------\n');
                    %----------

Transship(NoFac,NoPl,NoCD,td,su,dp,NoPr,pc,fd,dc,radius,d);
                    for m=1:NoPr
                        if sum(opc(:,m))~=sum(Y,2)'*dp(:,m)
                            fprintf('m: %g\n',m);
                            fprintf('sum(opc(:,m)): %g\n',sum(opc(:,m)));
                            fprintf('sum(Y,2)*dp(:,m):
%g\n',sum(Y,2)'*dp(:,m));
                            error('Production does not match demand');
                        end
                    end
                end

                if (j2>NoPl)&&(j2<=NoPl+NoCD)
                    list(1+length(list))=j2;
                end
            end

            if j4==length(list)
                j4=1;
            else
                j4=j4+1;
            end

        end

        nn=nn+1;

        if nn==rounds*iter
            figure
            subplot(3,2,1);

plot(xx(1:NoPl),yy(1:NoPl),'s','MarkerFaceColor','k','MarkerSize',9)
            axis equal
            axis([0 A 0 B])
            hold on

plot(xx(NoPl+1:NoPl+NoCD),yy(NoPl+1:NoPl+NoCD),'^','MarkerFaceColor','r',
'MarkerSize',8)
            hold on

plot(xx(NoPl+NoCD+1:NoFac),yy(NoPl+NoCD+1:NoFac),'^','MarkerSize',6)
            hold on

plot(xx(NoFac+1:NoFac+NoCus),yy(NoFac+1:NoFac+NoCus),'o','MarkerSize',2)
            hold off

            subplot(3,2,2);

plot(xx(1:NoPl),yy(1:NoPl),'s','MarkerFaceColor','k','MarkerSize',9)
```

```
            axis equal
            axis([0 A 0 B])
            hold on
            for j=NoPl+1:NoPl+NoCD
                if Z(j)==1

plot(xx(j),yy(j),'^','MarkerFaceColor','k','MarkerSize',8)
                end
                hold on
            end
            for j=NoPl+NoCD+1:NoFac
                if Z(j)==1
                    plot(xx(j),yy(j),'^','MarkerSize',6)
                end
                hold on
            end
            for i=NoFac+1:NoFac+NoCus
                if sum(Y(i,:))==1
                    plot(xx(i),yy(i),'o','MarkerSize',2)
                else

plot(xx(i),yy(i),'o','MarkerFaceColor','r','MarkerSize',2)
                end
                hold on
            end
            for i=NoFac+1:NoFac+NoCus
                for j=1:NoFac
                    if Y(i,j)==1
                        plot(xx([i,j]),yy([i,j]))
                        hold on
                    end
                end
            end
            hold off

            subplot(3,2,3);

plot(xx(1:NoPl),yy(1:NoPl),'s','MarkerFaceColor','k','MarkerSize',9)
            axis equal
            axis([0 A 0 B])
            hold on
            for j=NoPl+1:NoPl+NoCD
                if Z(j)==1

plot(xx(j),yy(j),'^','MarkerFaceColor','k','MarkerSize',8)
                    text(xx(j)+5,yy(j)+5,j,int2str(j),'FontSize',8);
                end
                hold on
            end
            for j=NoPl+NoCD+1:NoFac
                if Z(j)==1
                    plot(xx(j),yy(j),'^','MarkerSize',6)
                    text(xx(j)+5,yy(j)+5,j,int2str(j),'FontSize',8);
                end
                hold on
            end
```

```
            for j1=1:NoPl+NoCD
                for j2=1:NoFac
                    if U(j1,j2)>0
                        plot(xx([j1,j2]),yy([j1,j2]))
                        hold on
                    end
                end
            end
            hold off
        end

        if sum(sum(Y))<NoCus
            for i=NoFac+1:NoFac+NoCus
                if sum(Y(i,:))==0
                    td6=td(i,1:NoFac);
                    temp=1;
                    while temp==1
                        [~,j2]=min(td6(:));
                        if Z(j2)==0
                            Z(j2)=1;
                            if menu4==2
                                tZ(j2)=1;
                            end
                            break
                        else
                            td6(j2)=inf;
                        end
                    end
                    td7=td(j2,:);
                    td7(1:NoFac)=inf;
                    while min(td7)<=radius
                        [~,i1]=min(td7);
                        if sum(Y(i1,:))==0
                            if d(i1)<=(dc(j2)-odc(j2))
                                Y(i1,j2)=1;
                                odc(j2)=odc(j2)+d(i1);
                            else
                                break
                            end
                        end
                        td7(i1)=inf;
                    end
                end
            end
        end

        if nn==rounds*iter
        subplot(3,2,4);

plot(xx(1:NoPl),yy(1:NoPl),'s','MarkerFaceColor','k','MarkerSize',9)
            axis equal
            axis([0 A 0 B])
            hold on
            for j=NoPl+1:NoPl+NoCD
                if Z(j)==1
```

```
plot(xx(j),yy(j),'^','MarkerFaceColor','k','MarkerSize',8)
                end
                hold on
            end
            for j=NoPl+NoCD+1:NoFac
                if Z(j)==1
                    plot(xx(j),yy(j),'^','MarkerSize',6)
                end
                hold on
            end
            for i=NoFac+1:NoFac+NoCus
                if sum(Y(i,:))==1
                    plot(xx(i),yy(i),'o','MarkerSize',2)
                else

plot(xx(i),yy(i),'o','MarkerFaceColor','r','MarkerSize',2)
                end
                hold on
            end
            for i=NoFac+1:NoFac+NoCus
                for j=1:NoFac
                    if Y(i,j)==1
                        plot(xx([i,j]),yy([i,j]))
                        hold on
                    end
                end
            end
            hold off

             subplot(3,2,5);

plot(xx(1:NoPl),yy(1:NoPl),'s','MarkerFaceColor','k','MarkerSize',9)
            axis equal
            axis([0 A 0 B])
            hold on
            for j=NoPl+1:NoPl+NoCD
                if Z(j)==1

plot(xx(j),yy(j),'^','MarkerFaceColor','k','MarkerSize',8)
                    text(xx(j)+5,yy(j)+5,j,int2str(j),'FontSize',8);
                end
                hold on
            end
            for j=NoPl+NoCD+1:NoFac
                if Z(j)==1
                    plot(xx(j),yy(j),'^','MarkerSize',6)
                    text(xx(j)+5,yy(j)+5,j,int2str(j),'FontSize',8);
                end
                hold on
            end
            for j1=1:NoPl+NoCD
                for j2=1:NoFac
                    if U(j1,j2)>0
                        plot(xx([j1,j2]),yy([j1,j2]))
                        hold on
```

```
            end
        end
    end
    hold off
end

if menu4==1
    while sum(sum(U1))<(sum(Z)-NoPl)
        j2=NoPl+1;
        while j2<=NoFac
            if (Z(j2)==1)&&(sum(U1(:,j2))==0)
                temp=1;
                td8=td(j2,1:NoPl+NoCD);
                if j2<=NoPl+NoCD
                    td8(j2)=inf;
                end
                if min(td8(1:NoPl))<=fd
                    [~,j3]=min(td8(1:NoPl));
                    U1(j3,j2)=1;
                    U(j3,j2)=U(j3,j2)+d*Y(:,j2);
                    disp(j2);
                    disp(j3);
                    j2=j2+1;
                    continue
                end
                while min(td8)<=fd
                    [~,j3]=min(td8);
                    if (Z(j3)==1)&&(sum(U1(:,j3))==0)
                        temp=0;
                        td8(j3)=inf;
                    end
                    if (Z(j3)==1)&&(sum(U1(:,j3))==1)
                        U1(j3,j2)=1;
                        U(j3,j2)=U(j3,j2)+d*Y(:,j2);
                        disp(j2);
                        disp(j3);
                        j5=j3;
                        temp=1;
                        while temp==1
                            [j6,~]=find(U1(:,j5)==1);
                            U(j6,j5)=U(j6,j5)+d*Y(:,j2);
                            disp(j5);
                            disp(j6);
                            if j6<=NoPl
                                break
                            else
                                j5=j6;
                            end
                        end
                        temp=0;
                        break
                    else
                        td8(j3)=inf;
                    end
                end
                if temp==0
```

```
                        j2=j2+1;
                        continue
                    end
                    td8=td(j2,1:NoPl+NoCD);
                    if j2<=NoPl+NoCD
                        td8(j2)=inf;
                    end
                    [~,j3]=min(td8);
                    Z(j3)=1;
                    disp(j2);
                    disp(j3);
                    j2=j2+1;
                else
                    j2=j2+1;
                end
            end
        end
    else
        if sum(tZ)>0
            [j2]=find(tZ==1);
            jj=j2(1);

Transship(NoFac,NoPl,NoCD,td,su,dp,NoPr,pc,fd,dc,radius,d);
            for m=1:NoPr
                if sum(opc(:,m))~=sum(Y,2)'*dp(:,m)
                    fprintf('m: %g\n',m);
                    fprintf('sum(opc(:,m)): %g\n',sum(opc(:,m)));
                    fprintf('sum(Y,2)'*dp(:,m):
%g\n',sum(Y,2)'*dp(:,m));
                    error('Production does not match demand');
                end
            end
        end
    end

    if nn==rounds*iter
        subplot(3,2,6);

plot(xx(1:NoPl),yy(1:NoPl),'s','MarkerFaceColor','k','MarkerSize',9)
        axis equal
        axis([0 A 0 B])
        hold on
        for j=NoPl+1:NoPl+NoCD
            if Z(j)==1

plot(xx(j),yy(j),'^','MarkerFaceColor','k','MarkerSize',8)
                text(xx(j)+5,yy(j)+5,j,int2str(j),'FontSize',8);
            end
            hold on
        end
        for j=NoPl+NoCD+1:NoFac
            if Z(j)==1
                plot(xx(j),yy(j),'^','MarkerSize',6)
                text(xx(j)+5,yy(j)+5,j,int2str(j),'FontSize',8);
            end
            hold on
```

```
        end
        for j1=1:NoPl+NoCD
            for j2=1:NoFac
                if U(j1,j2)>0
                    plot(xx([j1,j2]),yy([j1,j2]))
                    hold on
                end
            end
        end
        hold off
    end



    if sum(sum(Y))<NoCus
        error('"More depot capacity is needed. Not all customers have
been assigned."');
    end

    if menu4==2
        U(1:NoFac+NoCus,1:NoFac+NoCus)=0;
        for j1=1:NoFac
            for j2=1:NoFac
                for m=1:NoPr
                    U(j1,j2)=U(j1,j2)+su(m)*UU(j1,j2,m);
                end
            end
        end
    end

    U2=U;
    Z2=Z;
    Y2=Y;
    if menu4==2
        UU2=UU;
    end
    %-------------------------



    CostEstimate(NoFac,Z,Y,td,U,o,tc,sc,d,vc,ft);

    fprintf('Depot Cost: %g\n\t\t',DepCost);
    fprintf('Estimated Tour Cost: %g',EstRoutCost);
    fprintf(' (%g',EstFixRoutCost);
    fprintf(',%g)\n\t\t',EstVarRoutCost);
    fprintf('Transshipment Cost: %g\n\t\t',TransCost);
    fprintf('Total Cost: %g\n\n\t\t',Cost);


    Overall{5,nn}=Cost;

    for j=1:NoFac
        if Z(j)==1
```

```
                freq1(j)=freq1(j)+1;
            end
        end

        %----End of Construction Phase GRASP-------------------------------
-------------

        %----Local Search Phase of GRASP-----------------------------------
----------

        temp=1;
        while temp==1
            temp=0;

            %--------Transshipment Path Swap--------
            if menu4==1
                for j=NoPl+1:NoFac
                    if sum(U(1:NoPl,j))>0
                        [j1,~]=find(U1(1:NoPl,j)==1);
                        if td(j1,j)>min(td(1:NoPl,j))
                            [j3,~]=find(td(1:NoPl,j)==min(td(1:NoPl,j)));
                            U1(j1,j)=0;
                            U1(j3,j)=1;
                            U(j3,j)=U(j1,j);
                            U(j1,j)=0;
                            fprintf('Transshipment Path Swap (Type
1)\n');
                            CostEstimate(NoFac,Z,Y,td,U,o,tc,sc,d,vc,ft);
                            fprintf('Total Cost: %g\n',Cost);
                            temp=1;
                        end
                    end
                end

                td5=td(1:NoFac,1:NoFac);
                for j=1:NoFac
                    if Z(j)==0
                        td5(:,j)=inf;
                    end
                    td5(j,j)=inf;
                end
                for j1=NoPl+1:NoFac
                    if
(Z(j1)==1)&&(sum(U(:,j1))>0)&&(sum(U(1:NoPl,j1))==0)
                        ttd1=0;
                        j3=j1;
                        j4=inf;
                        while j4>NoPl
                            [j4,~]=find(U1(:,j3)==1);
                            ttd1=ttd1+td(j4,j3);
                            j3=j4;
                        end
                        for j2=1:NoPl+NoCD
```

```
                                if
(td5(j1,j2)<=fd)&&(U1(j2,j1)==0)&&(U1(j1,j2)==0)
                                    ttd2=td(j2,j1);
                                    if j2>NoPl
                                        j3=j2;
                                        j4=inf;
                                        while j4>NoPl
                                            [j4,~]=find(U1(:,j3)==1);
                                            ttd2=ttd2+td(j4,j3);
                                            j3=j4;
                                        end
                                    end
                                    if ttd2<ttd1
                                        [j5,~]=find(U1(:,j1)==1);
                                        U1(j5,j1)=0;
                                        U1(j2,j1)=1;
                                        j3=j5;
                                        j4=inf;
                                        while j4>NoPl
                                            [j4,~]=find(U1(:,j3)==1);
                                            U(j4,j3)=U(j4,j3)-U(j5,j1);
                                            j3=j4;
                                        end
                                        if j2>NoPl
                                            j3=j2;
                                            j4=inf;
                                            while j4>NoPl
                                                [j4,~]=find(U1(:,j3)==1);
                                                U(j4,j3)=U(j4,j3)+U(j5,j1);
                                                j3=j4;
                                            end
                                        end
                                        U(j2,j1)=U(j5,j1);
                                        U(j5,j1)=0;
                                        fprintf('Transshipment Path Swap
(Type 2)\n');

CostEstimate(NoFac,Z,Y,td,U,o,tc,sc,d,vc,ft);
                                        fprintf('Total Cost: %g\n',Cost);
                                        temp=1;
                                    end
                                end
                            end
                        end
                    end
                else
                    for m=1:NoPr
                        for j=1:NoFac
                            if sum(UU(:,j,m))>0
                                [j1,~,~]=find(UU(:,j,m)>0);
                                for nn1=1:length(j1)
                                    [j2,~,~]=find(UU(:,j1(nn1),m)>0);
                                    for nn2=1:length(j2)
                                        if
UU(j2(nn2),j1(nn1),m)>=UU(j1(nn1),j,m)
```

```
ttd1=td(j,j1(nn1))+td(j1(nn1),j2(nn2));

[~,j3]=find(td(j,1:NoPl+NoCD)<=fd);
                                                for nn3=1:length(j3)
                                                    if
(j3(nn3)==j2(nn2))&&(j3(nn3)~=j)

UU(j2(nn2),j,m)=UU(j2(nn2),j,m)+UU(j1(nn1),j,m);

U(j2(nn2),j)=U(j2(nn2),j)+su(m)*UU(j1(nn1),j,m);


UU(j2(nn2),j1(nn1),m)=UU(j2(nn2),j1(nn1),m)-UU(j1(nn1),j,m);

U(j2(nn2),j1(nn1))=U(j2(nn2),j1(nn1))-su(m)*UU(j1(nn1),j,m);


U(j1(nn1),j)=U(j1(nn1),j)-su(m)*UU(j1(nn1),j,m);

odc(j1(nn1))=odc(j1(nn1))-su(m)*UU(j1(nn1),j,m);
                                                        UU(j1(nn1),j,m)=0;


fprintf('Transshipment Path Swap (type 1)\n');

CostEstimate(NoFac,Z,Y,td,U,o,tc,sc,d,vc,ft);
                                                        fprintf('m: %g\t',m);
                                                        fprintf('j: %g\t',j);
                                                        fprintf('j1:
%g\t',j1(nn1));
                                                        fprintf('j2:
%g\t',j2(nn2));
                                                        fprintf('j3:
%g\t',j3(nn3));
                                                        fprintf('Total Cost:
%g\n',Cost);
                                                        temp=1;
                                                        break
                                                    end

                                                    if
(j3(nn3)~=j)&&(j3(nn3)~=j1(nn1))&&(Z(j3(nn3))==1)&&(td(j3(nn3),j2(nn2))<=
fd)&&((dc(j3(nn3))-odc(j3(nn3)))>=(su(m)*UU(j1(nn1),j,m)))

ttd2=td(j,j3(nn3))+td(j3(nn3),j2(nn2));
                                                        if ttd2<ttd1

UU(j3(nn3),j,m)=UU(j3(nn3),j,m)+UU(j1(nn1),j,m);

U(j3(nn3),j)=U(j3(nn3),j)+su(m)*UU(j1(nn1),j,m);

odc(j3(nn3))=odc(j3(nn3))+su(m)*UU(j1(nn1),j,m);
```

```
UU(j2(nn2),j3(nn3),m)=UU(j2(nn2),j3(nn3),m)+UU(j1(nn1),j,m);
U(j2(nn2),j3(nn3))=U(j2(nn2),j3(nn3))+su(m)*UU(j1(nn1),j,m);

UU(j2(nn2),j1(nn1),m)=UU(j2(nn2),j1(nn1),m)-UU(j1(nn1),j,m);
U(j2(nn2),j1(nn1))=U(j2(nn2),j1(nn1))-su(m)*UU(j1(nn1),j,m);

U(j1(nn1),j)=U(j1(nn1),j)-su(m)*UU(j1(nn1),j,m);
odc(j1(nn1))=odc(j1(nn1))-su(m)*UU(j1(nn1),j,m);
UU(j1(nn1),j,m)=0;

fprintf('Transshipment Path Swap (type 2)\n');
CostEstimate(NoFac,Z,Y,td,U,o,tc,sc,d,vc,ft);
                                                    fprintf('m:
%g\t',m);
                                                    fprintf('j:
%g\t',j);
                                                    fprintf('j1:
%g\t',j1(nn1));
                                                    fprintf('j2:
%g\t',j2(nn2));
                                                    fprintf('j3:
%g\t',j3(nn3));
                                                    fprintf('Cost:
Cost: %g\n',Cost);
                                                    temp=1;
                                                    break
                                                end
                                            end
                                        end
                                    end
                                    if temp==1
                                        break
                                    end
                                end
                            end
                        end
                    if temp==1
                        break
                    end
                end
            end
        end
    end

    %---  --Customer reallocation-----------

    if menu4==1
```

```
                td3=td;
                td3(:,NoFac+1:NoFac+NoCus)=inf;
                td3(:,1:NoPl)=inf;
                for j=NoPl+1:NoFac
                    if Z(j)==0
                        td3(:,j)=inf;
                    end
                end
                for i=NoFac+1:NoFac+NoCus
                    [~,j1]=find(Y(i,1:NoFac)==1);
                    [~,j2]=min(td3(i,:));
                    if (j1~=j2)&&(j1>NoPl)&&(j2>NoPl)&&((dc(j2)-
odc(j2))>=d(i))
                        ttd1=0;
                        ttd2=0;
                        j3=j1;
                        j4=inf;
                        while j4>NoPl
                            [j4,~]=find(U1(:,j3)==1);
                            ttd1=ttd1+td(j4,j3);
                            j3=j4;
                        end
                        j3=j2;
                        j4=inf;
                        while j4>NoPl
                            [j4,~]=find(U1(:,j3)==1);
                            ttd2=ttd2+td(j4,j3);
                            j3=j4;
                        end
                        if ttd2<ttd1
                            Y(i,j1)=0;
                            odc(j1)=odc(j1)-d(i);
                            Y(i,j2)=1;
                            odc(j2)=odc(j2)+d(i);
                            j3=j1;
                            j4=inf;
                            while j4>NoPl
                                [j4,~]=find(U1(:,j3)==1);
                                U(j4,j3)=U(j4,j3)-d(i);
                                j3=j4;
                            end
                            j3=j2;
                            j4=inf;
                            while j4>NoPl
                                [j4,~]=find(U1(:,j3)==1);
                                U(j4,j3)=U(j4,j3)+d(i);
                                j3=j4;
                            end
                            fprintf('Customer Reallocation\n');
                            CostEstimate(NoFac,Z,Y,td,U,o,tc,sc,d,vc,ft);
                            fprintf('Total Cost: %g\n',Cost);
                            temp=1;
                        end
                    end
                end
            end
```

```
            %-------Depot Drop  - ---------
            if menu4==1
                for j=NoPl+1:NoFac
                    if (Z(j)==1)&&(sum(Y(:,j))==0)&&(sum(U1(j,:))==0)
                        Z(j)=0;
                        U1(:,j)=0;
                        fprintf('Depot Drop\n');
                        CostEstimate(NoFac,Z,Y,td,U,o,tc,sc,d,vc,ft);
                        fprintf('Total Cost: %g\n',Cost);
                        temp=1;
                    end
                end
            else
                for j=NoPl+1:NoFac
                    if
(Z(j)==1)&&(sum(Y(:,j))==0)&&(sum(sum(UU(j,:,:)))==0)
                        Z(j)=0;
                        fprintf('Depot Drop\n');
                        CostEstimate(NoFac,Z,Y,td,U,o,tc,sc,d,vc,ft);
                        fprintf('Total Cost: %g\n',Cost);
                        temp=1;
                    end
                end
            end
        end



        %- --- ---- ------  ---
        if menu4==2
            for j=NoPl+1:NoFac
                if (Z(j)==1)&&(sum(Y(:,j))>=1)
                    for m=1:NoPr
                        if sum(UU(:,j,m))==0
                            error('Not Linked');
                        end
                    end
                end
            end
        end


        %--- ---------- ----------


        CE=CostEstimate(NoFac,Z,Y,td,U,o,tc,sc,d,vc,ft);

        fprintf('Depot Cost: %g\n\n',DepCost);
        fprintf('Estimated Tour Cost: %g',EstRoutCost);
        fprintf(' (%g',EstFixRoutCost);
        fprintf(',%g)\n\t\t',EstVarRoutCost);
        fprintf('Transshipment Cost: %g\n\n',TransCost);
        fprintf('Total Cost: %g \n\n',Cost);
```

```
%----------------------


%----Updating Elite Solutions----------------------

if ee<=EliteSize
    Elite{1,ee}=rr;
    Elite{2,ee}=ii;
    Elite{3,ee}=Cost;
    Elite{4,ee}=Z;
    Elite{5,ee}=Y;
    Elite{6,ee}=U;
    Elite{7,ee}=U2;
    Elite{8,ee}=Z2;
    Elite{9,ee}=Y2;
    Elite{10,ee}=odc;
    if menu4==1
        Elite{11,ee}=U1;
    else
        Elite{11,ee}=UU;
    end
    if menu4==2
        Elite{12,ee}=UU2;
    end
    ee=ee+1;
elseif Cost<max([Elite{3,:}])
    [~,j2]=find([Elite{3,:}]==max([Elite{3,:}]));
    Elite{1,j2}=rr;
    Elite{2,j2}=ii;
    Elite{3,j2}=Cost;
    Elite{4,j2}=Z;
    Elite{5,j2}=Y;
    Elite{6,j2}=U;
    Elite{7,j2}=U2;
    Elite{8,j2}=Z2;
    Elite{9,j2}=Y2;
    Elite{10,j2}=odc;
    if menu4==1
        Elite{11,j2}=U1;
    else
        Elite{11,j2}=UU;
    end
    if menu4==2
        Elite{12,j2}=UU2;
    end
end

%----Diversification and Intensification----


Overall{1,nn}=rr;
Overall{2,nn}=ii;
Overall{3,nn}=Strategy;
Overall{4,nn}=Cost;
```

```
        Overall{6,nn}=100*(Overall{5,nn}-Overall{4,nn})/Overall{5,nn};

    end

end

%----Selecting best solution-----------------------------

[~,n]=find([Elite{3,:}]==min([Elite{3,:}]));
Z=Elite{4,n};
Y=Elite{5,n};
U=Elite{6,n};
U2=Elite{7,n};
Z2=Elite{8,n};
Y2=Elite{9,n};
odc=Elite{10,n};
if menu4==1
    U1=Elite{11,n};
else
    UU=Elite{11,n};
end
if menu4==2
    UU2=Elite{12,n};
end


if menu4==2
    for m=1:NoPr
        figure
        subplot(2,2,1);
        for j=1:NoPl
            plot(xx(j),yy(j),'s','MarkerFaceColor','k','MarkerSize',9)
            text(xx(j)-5,yy(j)+15,j,int2str(j),'FontSize=',8);
            text(xx(j)-10,yy(j)-
15,sum(UU2(j,:,m)),int2str(sum(UU2(j,:,m))),'FontSize=',8);
            hold on
        end

        axis equal
        axis([0 A 0 B])
        title(['Product ',num2str(m)]);
        hold on
        for j=NoPl+1:NoPl+NoCD
            if Z2(j)==1

plot(xx(j),yy(j),'^','MarkerFaceColor','k','MarkerSize',8)
                text(xx(j)-5,yy(j)+15,j,int2str(j),'FontSize',8);
                text(xx(j)-10,yy(j)-
15,sum(UU2(j,:,m)),int2str(sum(UU2(j,:,m))),'FontSize',8);

            end
            hold on
        end
        for j=NoPl+NoCD+1:NoFac
```

```
            if Z2(j)==1
                plot(xx(j),yy(j),'^','MarkerSize',6)
                text(xx(j)+5,yy(j)+5,j,int2str(j),'FontSize',8);
            end
            hold on
        end
        for j1=1:NoPl+NoCD
            for j2=1:NoFac
                if UU2(j1,j2,m)>0
                    plot(xx([j1,j2]),yy([j1,j2]));
                    hold on
                    xx1=xx(j1);
                    xx2=xx(j2);
                    yy1=yy(j1);
                    yy2=yy(j2);
                    Arrow(xx1,xx2,yy1,yy2);
                    plot([xx(j2),s],[yy(j2),t]);
                    plot([xx(j2),u],[yy(j2),v]);
                end
            end
        end

        subplot(2,2,3);
        for j=1:NoPl
            plot(xx(j),yy(j),'s','MarkerFaceColor','k','MarkerSize',9)
            text(xx(j)+6,yy(j)+13,j,int2str(j),'FontSize',10);
            hold on
        end

        axis equal
        axis([0 A 0 B])
        title(['Product ',num2str(m)]);
        hold on
        for j=NoPl+1:NoPl+NoCD
            if Z2(j)==1

plot(xx(j),yy(j),'^','MarkerFaceColor','k','MarkerSize',8)

            end
            hold on
        end
        for j=NoPl+NoCD+1:NoFac
            if Z2(j)==1
                plot(xx(j),yy(j),'^','MarkerSize',6)
            end
            hold on
        end
        for j1=1:NoPl+NoCD
            for j2=1:NoFac
                if UU2(j1,j2,m)>0
                    plot(xx([j1,j2]),yy([j1,j2]));
                    hold on
                    xx1=xx(j1);
                    xx2=xx(j2);
                    yy1=yy(j1);
```

```
                    yy2=yy(j2);
                    Arrow(xx1,xx2,yy1,yy2);
                    plot([xx(j2),s],[yy(j2),t]);
                    plot([xx(j2),u],[yy(j2),v]);
                end
            end
        end

        subplot(2,2,2);
        for j=1:NoPl
            plot(xx(j),yy(j),'s','MarkerFaceColor','k','MarkerSize',9)
            text(xx(j)-5,yy(j)+15,j,int2str(j),'FontSize',8);
            text(xx(j)-10,yy(j)-
15,sum(UU(j,:,m)),int2str(sum(UU(j,:,m))),'FontSize',8);
            hold on
        end

        axis equal
        axis([0 A 0 B])
        title(['Product ',num2str(m)]);
        hold on
        for j=NoPl+1:NoPl+NoCD
            if Z(j)==1

plot(xx(j),yy(j),'^','MarkerFaceColor','k','MarkerSize',8)
                text(xx(j)-5,yy(j)+15,j,int2str(j),'FontSize',8);
                text(xx(j)-10,yy(j)-
15,sum(UU(j,:,m)),int2str(sum(UU(j,:,m))),'FontSize',8);

            end
            hold on
        end
        for j=NoPl+NoCD+1:NoFac
            if Z(j)==1
                plot(xx(j),yy(j),'^','MarkerSize',6)
                text(xx(j)+5,yy(j)+5,j,int2str(j),'FontSize',8);
            end
            hold on
        end
        for j1=1:NoPl+NoCD
            for j2=1:NoFac
                if UU(j1,j2,m)>0
                    plot(xx([j1,j2]),yy([j1,j2]));
                    hold on
                    xx1=xx(j1);
                    xx2=xx(j2);
                    yy1=yy(j1);
                    yy2=yy(j2);
                    Arrow(xx1,xx2,yy1,yy2);
                    plot([xx(j2),s],[yy(j2),t]);
                    plot([xx(j2),u],[yy(j2),v]);
                end
            end
        end
```

```
        subplot(2,2,4);
        for j=1:NoPl
            plot(xx(j),yy(j),'s','MarkerFaceColor','k','MarkerSize',9)
            text(xx(j)+6,yy(j)+13,j,int2str(j),'FontSize',10);
            hold on
        end

        axis equal
        axis([0 A 0 B])
        title(['Product ',num2str(m)]);
        hold on
        for j=NoPl+1:NoPl+NoCD
            if Z(j)==1

plot(xx(j),yy(j),'^','MarkerFaceColor','k','MarkerSize',8)
            end
            hold on
        end
        for j=NoPl+NoCD+1:NoFac
            if Z(j)==1
                plot(xx(j),yy(j),'^','MarkerSize',6)
            end
            hold on
        end
        for j1=1:NoPl+NoCD
            for j2=1:NoFac
                if UU(j1,j2,m)>0
                    plot(xx([j1,j2]),yy([j1,j2]));
                    hold on
                    xx1=xx(j1);
                    xx2=xx(j2);
                    yy1=yy(j1);
                    yy2=yy(j2);
                    Arrow(xx1,xx2,yy1,yy2);
                    plot([xx(j2),s],[yy(j2),t]);
                    plot([xx(j2),u],[yy(j2),v]);
                end
            end
        end
    end
    hold off
end

fprintf('Best Solution Total Cost: %g\n\n',Elite{3,n});
figure
subplot(3,2,1);

plot(xx(1:NoPl),yy(1:NoPl),'s','MarkerFaceColor','k','MarkerSize',9)
axis equal
axis([0 A 0 B])
hold on
plot(xx(NoPl+1:NoPl+NoCD),yy(NoPl+1:NoPl+NoCD),'^','MarkerFaceColor','k',
'MarkerSize',8)
hold on
plot(xx(NoPl+NoCD+1:NoFac),yy(NoPl+NoCD+1:NoFac),'^','MarkerSize',6)
```

```
for j1=1:NoPl+NoCD
    for j2=1:NoFac
        if U2(j1,j2)>0
            plot(xx([j1,j2]),yy([j1,j2]))
            hold on
        end
    end
end
title('Transshipment paths after construction phase of GRASP');
hold off

subplot(3,2,4);
plot(xx(1:NoPl),yy(1:NoPl),'s','MarkerFaceColor','k','MarkerSize',9)
axis equal
axis([0 A 0 B])
hold on
for j=NoPl+1:NoPl+NoCD
    if Z(j)==1
        plot(xx(j),yy(j),'^','MarkerFaceColor','k','MarkerSize',8)
    end
    hold on
end
for j=NoPl+NoCD+1:NoFac
    if Z(j)==1
        plot(xx(j),yy(j),'^','MarkerSize',6)
    end
    hold on
end
for i=NoFac+1:NoFac+NoCus
    if sum(Y(i,:))==1
        plot(xx(i),yy(i),'o','MarkerSize',2)
    end
    hold on
end
for i=NoFac+1:NoFac+NoCus
    for j=1:NoFac
        if Y(i,j)==1
            plot(xx([i,j]),yy([i,j]))
            hold on
        end
    end
end
title('Allocation of customers after local search phase of GRASP');
hold off

subplot(3,2,5);
plot(xx(1:NoPl),yy(1:NoPl),'s','MarkerFaceColor','k','MarkerSize',9)
axis equal
axis([0 A 0 B])
hold on
for j=NoPl+1:NoPl+NoCD
    if Z(j)==1
        plot(xx(j),yy(j),'^','MarkerFaceColor','k','MarkerSize',8)
    end
    hold on
end
```

```
for j=NoPl+NoCD+1:NoFac
    if Z(j)==1
        plot(xx(j),yy(j),'^','MarkerSize',6)
    end
    hold on
end
for j1=1:NoPl+NoCD
    for j2=1:NoFac
        if U(j1,j2)>0
            plot(xx([j1,j2]),yy([j1,j2]))
            hold on
        end
    end
end
title('Transshipment paths after local search phase of GRASP');
hold off


%----Clarke Wright----------

CW=ClarkeWright(NoFac,NoCus,Z,Y,td,d,vc,tL);

%-------------------------

TourTravelCost=0;

for j1=1:NoFac+NoCus
    for j2=1:NoFac+NoCus
        TourTravelCost=TourTravelCost+tc*td(j1,j2)*sum(X(j1,j2,:));
    end
end

TransCost=0;

for j1=1:NoPl+NoCD
    for j2=1:NoFac
        TransCost=TransCost+sc*td(j1,j2)*U(j1,j2);
    end
end

DepCost=o*Z';
FixTourCost=ft*k;
TourCost=TourTravelCost+FixTourCost;
Cost=DepCost+TourCost+TransCost;

fprintf('Depot Cost: %g \n',DepCost);
fprintf('Tour Cost: %g',TourCost);
fprintf(' (Fixed Tour Cost: %g',FixTourCost);
fprintf(', Tour Travelling Cost: %g) \n',TourTravelCost);
fprintf('Transshipment Cost: %g \n',TransCost);
fprintf('Total Cost: %g \n\n',Cost);

%-------- Election ---------
```

```
    RE=RegoEjection(NoFac,NoCus,Z,Y,td,d,vc,tL);

%-------------------------- ... .. ---- -------

subplot(3,2,6);
plot(xx(1:NoPl),yy(1:NoPl),'s','MarkerFaceColor','k','MarkerSize',9)
axis equal
axis([0 A 0 B])
hold on
for j=NoPl+1:NoPl+NoCD
    if Z(j)==1
        plot(xx(j),yy(j),'^','MarkerFaceColor','k','MarkerSize',8)
    end
    hold on
end
for j=NoPl+NoCD+1:NoFac
    if Z(j)==1
        plot(xx(j),yy(j),'^','MarkerSize',6)
    end
    hold on
end
for i=NoFac+1:NoFac+NoCus
    if sum(Y(i,:))==1
        plot(xx(i),yy(i),'o','MarkerSize',2)
    end
    hold on
end
for k4=1:k;
    j6=1:sum(sum(X(:,:,k4)))+1;
    plot(xx(tour(k4,j6)),yy(tour(k4,j6)))
    hold on
end
title('Tours from facilities to customers');
hold off

%---- - - - -- --- ---- -

if menu4==2
    figure

    subplot(2,2,1);
    for j=1:NoPl
        plot(xx(j),yy(j),'s','MarkerFaceColor','k','MarkerSize',9)
        text(xx(j)+6,yy(j)+13,j,int2str(j),'FontSize',10);
        hold on
    end
    axis equal
    axis([0 A 0 B])
    hold on

plot(xx(NoPl+1:NoPl+NoCD),yy(NoPl+1:NoPl+NoCD),'^','MarkerFaceColor','k',
'MarkerSize',8)
    hold on
    plot(xx(NoPl+NoCD+1:NoFac),yy(NoPl+NoCD+1:NoFac),'^','MarkerSize',6)
    hold on
```

```
plot(xx(NoFac+1:NoFac+NoCus),yy(NoFac+1:NoFac+NoCus),'o','MarkerSize',2)
    title('Locations of facilities and customers');
    hold off

    subplot(2,2,2);
    plot(xx(1:NoPl),yy(1:NoPl),'s','MarkerFaceColor','k','MarkerSize',9)
    axis equal
    axis([0 A 0 B])
    hold on
    for j=NoPl+1:NoPl+NoCD
        if Z(j)==1
            plot(xx(j),yy(j),'^','MarkerFaceColor','k','MarkerSize',8)
        end
        hold on
    end
    for j=NoPl+NoCD+1:NoFac
        if Z(j)==1
            plot(xx(j),yy(j),'^','MarkerSize',6)
        end
        hold on
    end
    for i=NoFac+1:NoFac+NoCus
        if sum(Y(i,:))==1
            plot(xx(i),yy(i),'o','MarkerSize',2)
        end
        hold on
    end
    for i=NoFac+1:NoFac+NoCus
        for j=1:NoFac
            if Y(i,j)==1
                plot(xx([i,j]),yy([i,j]))
                hold on
            end
        end
    end
    title('Allocation of customers');
    hold off

    subplot(2,2,3);
    plot(xx(1:NoPl),yy(1:NoPl),'s','MarkerFaceColor','k','MarkerSize',9)
    axis equal
    axis([0 A 0 B])
    hold on
    for j=NoPl+1:NoPl+NoCD
        if Z(j)==1
            plot(xx(j),yy(j),'^','MarkerFaceColor','k','MarkerSize',8)
        end
        hold on
    end
    for j=NoPl+NoCD+1:NoFac
        if Z(j)==1
            plot(xx(j),yy(j),'^','MarkerSize',6)
        end
        hold on
    end
```

```
    for i=NoFac+1:NoFac+NoCus
        if sum(Y(i,:))==1
            plot(xx(i),yy(i),'o','MarkerSize',2)
        end
        hold on
    end
    for k4=1:k;
        j6=1:sum(sum(X(:,:,k4)))+1;
        plot(xx(tour(k4,j6)),yy(tour(k4,j6)))
        hold on
    end
    title('Tours from facilities to customers');
    hold off
end


%-------------------------------------

TourTravelCost=0;

for j1=1:NoFac+NoCus
    for j2=1:NoFac+NoCus
        TourTravelCost=TourTravelCost+tc*td(j1,j2)*sum(X(j1,j2,:));
    end
end

TransCost=0;

for j1=1:NoPl+NoCD
    for j2=1:NoFac
        TransCost=TransCost+sc*td(j1,j2)*U(j1,j2);
    end
end

DepCost=o*Z';
FixTourCost=ft*k;
TourCost=TourTravelCost+FixTourCost;
Cost=DepCost+TourCost+TransCost;

fprintf('Depot Cost: %g\n',DepCost);
fprintf('Tour Cost: %g',TourCost);
fprintf('  (Fixed Tour Cost: %g',FixTourCost);
fprintf(', Tour Travelling Cost: %g)\n',TourTravelCost);
fprintf('Transshipment Cost: %g\n',TransCost);
fprintf('Total Cost: %g\n\n',Cost);

toc;

for j=1:NoFac
    if Z(j)==1
        CapRatio(j)=odc(j)/dc(j);
    end
end
AveCapRatio=sum(CapRatio)/(sum(Z)-NoPl);
```

```
AveImprove=mean([Overall{6,1:nn}]);

figure
nn1=1;
while nn1<nn
    if strcmp('Unbiased',Overall{3,nn1})==1
        plot(nn1:nn1+iter-1,[Overall{4,nn1:nn1+iter-1}],':o')
    elseif strcmp('Intensification',Overall{3,nn1})==1
        plot(nn1:nn1+iter-1,[Overall{4,nn1:nn1+iter-1}],' *')
    elseif strcmp('Diversification',Overall{3,nn1})==1
        plot(nn1:nn1+iter-1,[Overall{4,nn1:nn1+iter-1}],'--s')
    end
    hold on
    nn1=nn1+iter;
end
xlabel('Iteration');
ylabel('Cost');
legend('Unbiased','Diversification','Intensification','location','Best')
hold off


figure
plot(1:nn,[Overall{6,1:nn}],'--s','MarkerSize',4)
xlabel('Iteration');
ylabel('Improvement Percentage');
hold off

menu2=menu('Save?','Yes','No');
if menu2==1
    if menu4==1

save('example.mat','A','B','NoVar','NoVar','NoPar','NoAl','NoAl','d','dr',
'dp','fd','radius','tb','it','c','sc','sc1','sc','sa','xx','xx','yy')
    else

save('example_H.mat','A','B','NoVar','NoVar','NoPar','NoAl','NoAl','d'
,'dr','dp','fd','radius','tb','it','c','sc','sa','xx','xx',
'yy')
    end
end
```

APPENDIX E: TRANSSHIPMENT FUNCTION

```
function TR=Transship(NoFac,NoPl,NoCD,td,su,dp,NoPr,pc,fd,dc,radius,d)

global Z UU opc odc tZ Y DP jj jj1 jj2 jj3

while sum(tZ)>0
    fprintf('
    fprintf('\njj: %g\n',jj);
    DP(1:NoFac)=0;
    for m=1:NoPr
        fprintf('\nm: %g\n',m);
        DP(jj)=sum(dp(:,m)'*Y(:,jj));
        fprintf('DP(jj): %g\n',DP(jj));
        td9=td(jj,1:NoPl);
        jj3=jj;
        while DP(jj3)>0.05
            [~,jj1]=min(td9);
            fprintf('jj1: %g\n',jj1);
            XX=min(DP(jj3),floor(pc(jj1,m)-opc(jj1,m)));
            if XX>0.05
                if jj1==jj3
                    opc(jj1,m)=opc(jj1,m)+XX;
                    if m==2
                        fprintf('opc: %g\n',opc(2,m));
                    end
                    DP(jj3)=DP(jj3)-XX;

                elseif td(jj1,jj3)<=fd
                    UU(jj1,jj3,m)=UU(jj1,jj3,m)+XX;
                    opc(jj1,m)=opc(jj1,m)+XX;
                    odc(jj1)=odc(jj1)+XX*su(m);
                    if m==2
                        fprintf('opc: %g\n',opc(2,m));
                    end
                    DP(jj3)=DP(jj3)-XX;
                else
                    XX2=XX;
                    while XX>0.05
                        jj2=0;
                        ttd=inf;
                        for j=1:NoPl+NoCD
                            if
(td(jj3,j)<=fd)&&(Z(j)==1)&&(jj3~=j)&&(((dc(j)-
odc(j))/su(m))>=1)&&(td(jj3,jj1)>td(j,jj1))
                                if td(j,jj1)<ttd
                                    ttd=td(j,jj1);
                                    jj2=j;
                                end
                            end
                        end
```

```
                        if jj2==0
                            for j=1:NoPl+NoCD
                                if
(td(jj3,j)<=fd)&&(Z(j)==0)&&(jj3~=j)&&(((dc(j)-odc(j))/su(m))>=1)
                                    if td(j,jj1)<ttd
                                        ttd=td(j,jj1);
                                        jj2=j;
                                    end
                                end
                            end
                            if jj2==0
                                error('not enough CDs for
transshipment');
                            end
                        end

                        XX1=min(XX,floor((dc(jj2)-odc(jj2))/su(m)));
                        UU(jj2,jj3,m)=UU(jj2,jj3,m)+XX1;
                        odc(jj2)=odc(jj2)+XX1*su(m);
                        fprintf('j
                        fprintf('
                        XX=XX-XX1;
                        fprintf('
                        fprintf('
                            DP(jj2)=DP(jj2)+XX1;

                        if Z(jj2)==0
                            Z(jj2)=1;
                            tZ(jj2)=1;
                        end

                    end
                    DP(jj3)=DP(jj3)-XX2;
                    fprintf('
                    fprintf('
                end
            end
            if DP(jj3)<=0.05
                if sum(DP)>0.05
                    [jj4]=find(DP>0.05);
                    jj3=jj4(1);
                    td9=td(jj3,1:NoPl);
                    fprintf('
                    fprintf('

                end
            else
                td9(jj1)=inf;
            end

        end
    end
    tZ(jj)=0;
```

```
    if sum(tZ)>0
        [j2]=find(tZ==1);
        j2=j2(1);
        td2=td(j2,:);
        td2(1:NoFac)=inf;

        while min(td2)<=radius
            [~,i1]=min(td2);
            if sum(Y(i1,:))==0
                if d(i1)<=(dc(j2)-odc(j2))
                    Y(i1,j2)=1;
                    odc(j2)=odc(j2)+d(i1);
                else
                    break
                end
            end
            td2(i1)=inf;
        end

        jj=j2;
    end
end

TR{1}=Z;
TR{2}=UU;
TR{3}=opc;
TR{4}=odc;
TR{5}=tZ;
TR{6}=Y;
TR{7}=DP;
```

APPENDIX F: COST ESTIMATE FUNCTION

```
function CE=CostEstimate(NoFac,Z,Y,td,U,o,tc,sc,d,vc,ft)

global EstFixRoutCost EstVarRoutCost EstRoutCost TransCost DepCost Cost

a=1.8;
b=1.8;

EstFixRoutCost=0;
EstVarRoutCost=0;

for j=1:NoFac
    if (Z(j)==1)&&(sum(Y(:,j))>0)
        ns=vc/(d*Y(:,j)/sum(Y(:,j)));
        DD=sum(td(:,j).*Y(:,j));
        EstFixRoutCost=EstFixRoutCost+ft*ceil(d*Y(:,j)/vc);

EstVarRoutCost=EstVarRoutCost+tc*(a*DD/ns+b*DD/(sum(Y(:,j))^0.5));
    end
end

EstRoutCost=EstFixRoutCost+EstVarRoutCost;

TransCost=sc*sum(sum(td.*U));

DepCost=o*Z';

Cost=DepCost+EstRoutCost+TransCost;

CE{1}=EstFixRoutCost;
CE{2}=EstVarRoutCost;
CE{3}=EstRoutCost;
CE{4}=TransCost;
CE{5}=DepCost;
CE{6}=Cost;
```

APPENDIX G: CLARKE WRIGHT FUNCTION

```
function CW=ClarkeWright(NoFac,NoCus,Z,Y,td,d,vc,tL)

global k X ovc tour CW L L2

X1(1:NoFac+NoCus,1:NoFac+NoCus,1)=0;

k3=0;

for j=1:NoFac
    if Z(j)==1
       disp(j);
       s(1:NoFac+NoCus,1:NoFac+NoCus)=-inf;

       for i3=NoFac+1:NoFac+NoCus-1
           for i4=i3+1:NoFac+NoCus
               if ((Y(i3,j)==1)&&(Y(i4,j)==1))
                   s(i3,i4)=td(i3,j)+td(i4,j)-td(i3,i4);
               end
           end
       end

       while max(max(s))>-inf

           [i3,i4]=find(s==max(max(s)));
           i5=i3(1);
           i6=i4(1);

           if
((sum(sum(X1(i5,:,:)))==0)&&(sum(sum(X1(i6,:,:)))==0)&&((d(i5)+d(i6))<=vc
))&&((td(j,i5)+td(i5,i6)+td(i6,j))<=tL)
              k3=k3+1;
              ovc1(k3)=0;
              X1(j,i5,k3)=1;
              X1(i5,i6,k3)=1;
              X1(i6,j,k3)=1;
              ovc1(k3)=ovc1(k3)+d(i5)+d(i6);
              L1(k3)=td(j,i5)+td(i5,i6)+td(i6,j);
           end

           if ((sum(X1(i5,j,:))==1)&&(sum(sum(X1(i6,:,:)))==0))
               for k4=1:k3
                   if X1(i5,j,k4)==1
                       k1=k4;
                       break
                   end
               end
               if (d(i6)<=(vc-ovc1(k1)))&&((L1(k1)-
td(i5,j)+td(i5,i6)+td(i6,j))<=tL)
```

```
                    X1(i5,j,k1)=0;
                    X1(i5,i6,k1)=1;
                    X1(i6,j,k1)=1;
                    ovc1(k1)=ovc1(k1)+d(i6);
                    L1(k1)=L1(k1)-td(i5,j)+td(i5,i6)+td(i6,j);
                end
            end

            if ((sum(X1(j,i5,:))==1)&&(sum(sum(X1(i6,:,:)))==0))
                for k4=1:k3
                    if X1(j,i5,k4)==1
                        k1=k4;
                        break
                    end
                end
                if (d(i6)<=(vc-ovc1(k1)))&&((L1(k1)-
td(j,i5)+td(j,i6)+td(i6,i5))<=tL)
                    X1(j,i5,k1)=0;
                    X1(j,i6,k1)=1;
                    X1(i6,i5,k1)=1;
                    ovc1(k1)=ovc1(k1)+d(i6);
                    L1(k1)=L1(k1)-td(j,i5)+td(j,i6)+td(i6,i5);
                end
            end

            if ((sum(sum(X1(i5,:,:)))==0)&&(sum(X1(i6,j,:))==1))
                for k4=1:k3
                    if X1(i6,j,k4)==1
                        k1=k4;
                        break
                    end
                end
                if (d(i5)<=(vc-ovc1(k1)))&&((L1(k1)-
td(i6,j)+td(i6,i5)+td(i5,j))<=tL)
                    X1(i6,j,k1)=0;
                    X1(i6,i5,k1)=1;
                    X1(i5,j,k1)=1;
                    ovc1(k1)=ovc1(k1)+d(i5);
                    L1(k1)=L1(k1)-td(i6,j)+td(i6,i5)+td(i5,j);
                end
            end

            if ((sum(sum(X1(i5,:,:)))==0)&&(sum(X1(j,i6,:))==1))
                for k4=1:k3
                    if X1(j,i6,k4)==1
                        k1=k4;
                        break
                    end
                end
                if (d(i5)<=(vc-ovc1(k1)))&&((L1(k1)-
td(j,i6)+td(j,i5)+td(i5,i6))<=tL)
                    X1(j,i6,k1)=0;
                    X1(j,i5,k1)=1;
                    X1(i5,i6,k1)=1;
                    ovc1(k1)=ovc1(k1)+d(i5);
```

```
                L1(k1)=L1(k1)-td(j,i6)+td(j,i5)+td(i5,i6);
            end
        end

        if ((sum(X1(i5,j,:))==1)&&(sum(X1(j,i6,:))==1))
            for k4=1:k3
                if X1(i5,j,k4)==1
                    k1=k4;
                    break
                end
            end
            for k4=1:k3
                if X1(j,i6,k4)==1
                    k2=k4;
                    break
                end
            end
            if ((k1~=k2)&&((ovc1(k1)+ovc1(k2))<=vc))&&((L1(k1)-
td(i5,j)+L1(k2)-td(j,i6)+td(i5,i6))<=tL)
                X1(i5,j,k1)=0;
                L1(k1)=L1(k1)-td(i5,j);
                X1(j,i6,k2)=0;
                L1(k2)=L1(k2)-td(j,i6);
                k3=k3+1;
                for j1=1:NoFac+NoCus
                    for j2=1:NoFac+NoCus
                        X1(j1,j2,k3)=X1(j1,j2,k1)+X1(j1,j2,k2);
                        X1(j1,j2,k1)=0;
                        X1(j1,j2,k2)=0;
                    end
                end
                X1(i5,i6,k3)=1;
                ovc1(k3)=ovc1(k1)+ovc1(k2);
                L1(k3)=L1(k1)+L1(k2)+td(i5,i6);
                ovc1(k1)=0;
                ovc1(k2)=0;
            end
        end

        if ((sum(X1(j,i5,:))==1)&&(sum(X1(i6,j,:))==1))
            for k4=1:k3
                if X1(j,i5,k4)==1
                    k1=k4;
                    break
                end
            end
            for k4=1:k3
                if X1(i6,j,k4)==1
                    k2=k4;
                    break
                end
            end
            if ((k1~=k2)&&((ovc1(k1)+ovc1(k2))<=vc))&&((L1(k1)-
td(j,i5)+L1(k2)-td(i6,j)+td(i6,i5))<=tL)
                X1(j,i5,k1)=0;
                L1(k1)=L1(k1)-td(j,i5);
```

```
                    X1(i6,j,k2)=0;
                    L1(k2)=L1(k2)-td(i6,j);
                    k3=k3+1;
                    for j1=1:NoFac+NoCus
                        for j2=1:NoFac+NoCus
                            X1(j1,j2,k3)=X1(j1,j2,k1)+X1(j1,j2,k2);
                            X1(j1,j2,k1)=0;
                            X1(j1,j2,k2)=0;
                        end
                    end
                    X1(i6,i5,k3)=1;
                    ovc1(k3)=ovc1(k1)+ovc1(k2);
                    L1(k3)=L1(k1)+L1(k2)+td(i6,i5);
                    ovc1(k1)=0;
                    ovc1(k2)=0;
                end
            end
            if ((sum(X1(i5,j,:))==1)&&(sum(X1(i6,j,:))==1))
                for k4=1:k3
                    if X1(i5,j,k4)==1
                        k1=k4;
                        break
                    end
                end
                for k4=1:k3
                    if X1(i6,j,k4)==1
                        k2=k4;
                        break
                    end
                end
                if ((k1~=k2)&&((ovc1(k1)+ovc1(k2))<=vc))&&((L1(k1)-
td(i5,j)+L1(k2)-td(i6,j)+td(i5,i6))<=tL)
                    X1(i5,j,k1)=0;
                    L1(k1)=L1(k1)-td(i5,j);
                    X1(i6,j,k2)=0;
                    L1(k2)=L1(k2)-td(i6,j);
                    k3=k3+1;
                    for j1=1:NoFac+NoCus
                        for j2=1:NoFac+NoCus
                            X1(j1,j2,k3)=X1(j1,j2,k1)+X1(j2,j1,k2);
                            X1(j1,j2,k1)=0;
                            X1(j2,j1,k2)=0;
                        end
                    end
                    X1(i5,i6,k3)=1;
                    ovc1(k3)=ovc1(k1)+ovc1(k2);
                    L1(k3)=L1(k1)+L1(k2)+td(i5,i6);
                    ovc1(k1)=0;
                    ovc1(k2)=0;
                end
            end
            if ((sum(X1(j,i5,:))==1)&&(sum(X1(j,i6,:))==1))
                for k4=1:k3
                    if X1(j,i5,k4)==1
                        k1=k4;
                        break
```

```
                end
            end
            for k4=1:k3
                if X1(j,i6,k4)==1
                    k2=k4;
                    break
                end
            end
            if ((k1~=k2)&&((ovc1(k1)+ovc1(k2))<=vc))&&((L1(k1)-
td(j,i5)+L1(k2)-td(j,i6)+td(i5,i6))<=tL)
                X1(j,i5,k1)=0;
                L1(k1)=L1(k1)-td(j,i5);
                X1(j,i6,k2)=0;
                L1(k2)=L1(k2)-td(j,i6);
                k3=k3+1;
                for j1=1:NoFac+NoCus
                    for j2=1:NoFac+NoCus
                        X1(j1,j2,k3)=X1(j2,j1,k1)+X1(j1,j2,k2);
                        X1(j2,j1,k1)=0;
                        X1(j1,j2,k2)=0;
                    end
                end
                X1(i5,i6,k3)=1;
                ovc1(k3)=ovc1(k1)+ovc1(k2);
                L1(k3)=L1(k1)+L1(k2)+td(i5,i6);
                ovc1(k1)=0;
                ovc1(k2)=0;
            end
        end
        s(i5,i6)=-inf;
    end

    for i3=NoFac+1:NoFac+NoCus
        if ((Y(i3,j)==1)&&(sum(sum(X1(i3,:,:)))==0))
            k3=k3+1;
            X1(i3,j,k3)=1;
            X1(j,i3,k3)=1;
            ovc1(k3)=d(i3);
            L1(k3)=td(i3,j)+td(j,i3);
        end
    end
  end
end

%-----------------------

k=0;
for k4=1:k3
    if sum(sum(X1(:,:,k4)))>0
        k=k+1;
        for j1=1:NoFac+NoCus
            for j2=1:NoFac+NoCus
                X(j1,j2,k)=X1(j1,j2,k4);
                ovc(k)=ovc1(k4);
                L(k)=L1(k4);
```

```
            end
        end
    end
end



k5=1;
L2(k5)=0;
for j=1:NoFac
    if Z(j)==1
        while sum(X(j,:,k5))>=1
            j3=1;
            tour(k5,j3)=j;
            j4=j;

            while j3<=sum(sum(X(:,:,k5)))
                [~,j5]=find(X(j4,:,k5)==1);
                j3=j3+1;
                tour(k5,j3)=j5;
                L2(k5)=L2(k5)+td(tour(k5,j3-1),j5);
                j4=j5;
            end
            if abs(L2(k5)-L(k5))>0.001
                error('"Wrong tour length"');
            end

            if k5==k
                break
            else
                k5=k5+1;
                L2(k5)=0;
            end
        end
    end
end

if max(L)>tL
    error('"Max tour length is exceeded"');
end

if max(ovc)>vc
    error('"Vehicle capacity is exceeded"');
end

CW{1}=k;
CW{2}=X;
CW{3}=ovc;
CW{4}=tour;
CW{5}=L;
CW{6}=L2;
```

APPENDIX H: EJECTION CHAIN FUNCTION

```
function RE=RegoEjection(NoFac,NoCus,Z,Y,td,d,vc,tL)

global k X ovc tour ej RE L

l=6;
h=25;



for j1=1:NoFac
    if sum(Y(:,j1))>0
        fprintf('Facility: %g\n',j1);

        tabu(NoFac+1:NoFac+NoCus,NoFac+1:NoFac+NoCus)=0;
        ne(NoFac+1:NoFac+NoCus,1:h)=0;

        td1=td;
        td1(1:NoFac+NoCus,1:NoFac)=inf;

        for i=NoFac+1:NoFac+NoCus
            if Y(i,j1)==1
                td1(i,i)=inf;
                n=1;
                while n<=h
                    if min(td1(i,:))==inf
                        break
                    end
                    [~,i1]=min(td1(i,:));
                    if Y(i1,j1)==1
                        ne(i,n)=i1;
                        n=n+1;
                    end
                    td1(i,i1)=inf;
                end
            end
        end

        for rr=1:10
            if mod(rr,100)==0
                disp(rr);
            end
            if mod(rr,2)==1
                type=1;
            else
                type=2;
            end
```

```
                for i1=NoFac+1:NoFac+NoCus
                    for i2=NoFac+1:NoFac+NoCus
                        if tabu(i1,i2)>0
                            tabu(i1,i2)=tabu(i1,i2)-1;
                        end
                    end
                end

                e(2)=inf;
                ovc1=ovc;
                L1=L;
                ej=[];
                v=[];

                for i1=NoFac+1:NoFac+NoCus
                    if Y(i1,j1)==1
                        [k1,nn1]=find(tour==i1);
                        ej(1,1)=tour(k1,nn1-1);
                        ej(1,2)=i1;
                        ej(1,3)=tour(k1,nn1+1);
                        temp=0;
                        for n=1:h
                            i2=ne(i1,n);
                            if (i2~=j1)&&(i2~=0)&&(isempty(find(ej==i2,
1))==1)&&(tabu(i1,i2)==0)
                                temp=1;
                                [k2,nn2]=find(tour==i2);
                                ej(2,1)=tour(k2,nn2-1);
                                ej(2,2)=i2;
                                ej(2,3)=tour(k2,nn2+1);
                                t1=td(ej(2,1),ej(1,2))+td(ej(1,2),ej(2,3));
                                t2=td(ej(1,1),ej(1,2))+td(ej(1,2),ej(1,3));
                                t3=td(ej(2,1),ej(2,2))+td(ej(2,2),ej(2,3));
                                t4=td(ej(1,1),ej(1,3));

                                e1(2)=t1-t2-t3;
                                e2(2)=t1-t2-t3+t4;
                                if (type==1)&&(e1(2)<e(2))
                                    e(2)=e1(2);
                                    v(1)=i1;
                                    v(2)=i2;
                                elseif (type==2)&&(e2(2)<e(2))
                                    e(2)=e2(2);
                                    v(1)=i1;
                                    v(2)=i2;

                                end
                            end
                        end
                    end
                end

                if temp==0
                    break
```

```
            end

            [k1,nn1]=find(tour==v(1));
            ej(1,1)=tour(k1,nn1-1);
            ej(1,2)=v(1);
            ej(1,3)=tour(k1,nn1+1);
            ovc1(k1)=ovc1(k1)-d(v(1));
            L1(k1)=L1(k1)-td(ej(1,1),ej(1,2))-td(ej(1,2),ej(1,3));
            if type==2
                L1(k1)=L1(k1)+td(ej(1,1),ej(1,3));
            end

            [k2,nn2]=find(tour==v(2));
            ej(2,1)=tour(k2,nn2-1);
            ej(2,2)=v(2);
            ej(2,3)=tour(k2,nn2+1);
            ovc1(k2)=ovc1(k2)-d(v(2))+d(v(1));
            L1(k2)=L1(k2)-td(ej(2,1),ej(2,2))-
td(ej(2,2),ej(2,3))+td(ej(2,1),ej(1,2))+td(ej(1,2),ej(2,3));

            delta=inf;
            m=2;
            m1=0;
            while (m<=l)

                if (type==1)
                    if m1>0
                        ovc1(k1)=ovc1(k1)-d(v(m1));
                        L1(k1)=L1(k1)-td(ej(1,1),ej(m1,2))-
td(ej(m1,2),ej(1,3));
                    end
                    ovc1(k1)=ovc1(k1)+d(v(m));

L1(k1)=L1(k1)+td(ej(1,1),ej(m,2))+td(ej(m,2),ej(1,3));
                    if (max(ovc1)<=vc)&&(max(L1)<=tL)

delta1(m)=e(m)+td(ej(1,1),ej(m,2))+td(ej(m,2),ej(1,3));
                        if delta1(m)<delta
                            delta=delta1(m);

                            m1=m;

                        else
                            if m1>0
                                ovc1(k1)=ovc1(k1)+d(v(m1));

L1(k1)=L1(k1)+td(ej(1,1),ej(m1,2))+td(ej(m1,2),ej(1,3));
                            end
                            ovc1(k1)=ovc1(k1)-d(v(m));
                            L1(k1)=L1(k1)-td(ej(1,1),ej(m,2))-
td(ej(m,2),ej(1,3));
                        end
                    else
                        if m1>0
                            ovc1(k1)=ovc1(k1)+d(v(m1));
```

```
L1(k1)=L1(k1)+td(ej(1,1),ej(m1,2))+td(ej(m1,2),ej(1,3));
                        end
                        ovc1(k1)=ovc1(k1)-d(v(m));
                        L1(k1)=L1(k1)-td(ej(1,1),ej(m,2))-
td(ej(m,2),ej(1,3));
                    end
                end

                if (type==2)
                    if (max(ovc1)<=vc)&&(max(L1)<=tL)
                        delta2=inf;
                        for p2=1:NoFac+NoCus
                            if (isempty(find(v==p2,
1))==1)&&(Y(p2,j1)==1)
                                for q2=1:NoFac+NoCus
                                    if
(sum(X(p2,q2,:)==1))&&(isempty(find(v==q2, 1))==1)&&(Y(q2,j1)==1)
                                        for k7=1:k
                                            if X(p2,q2,k7)==1
                                                k8=k7;
                                                break
                                            end
                                        end
                                        if
((ovc1(k8)+d(v(m)))<=vc)&&((L1(k8)+td(p2,v(m))+td(v(m),q2)-
td(p2,q2))<=tL)

delta3=td(p2,v(m))+td(v(m),q2)-td(p2,q2);
                                            if delta3<delta2
                                                delta2=delta3;
                                                p1=p2;
                                                q1=q2;
                                                k9=k8;
                                            end
                                        end
                                    end
                                end

                            end
                        end
                        delta1(m)=e(m)+delta2;

                        if delta1(m)<delta
                            delta=delta1(m);
                            m1=m;
                            p=p1;
                            q=q1;
                            k3=k9;

                        end
                    end
                end
```

```
                m=m+1;
                if m>l
                    m=m-1;
                    break

                end
                e(m)=inf;

                temp=0;

                for n=1:h
                    i2=ne(v(m-1),n);
                    if (i2~=j1)&&(i2~=0)&&(isempty(find(ej==i2, 1))==1)

                        temp=1;
                        [k2,nn2]=find(tour==i2);
                        ej(m,1)=tour(k2,nn2-1);
                        ej(m,2)=i2;
                        ej(m,3)=tour(k2,nn2+1);
                        t1=td(ej(m,1),ej(m-1,2))+td(ej(m-1,2),ej(m,3));
                        t3=td(ej(m,1),ej(m,2))+td(ej(m,2),ej(m,3));

                        e1(m)=e(m-1)+t1-t3;
                        e2(m)=e(m-1)+t1-t3;
                        if (type==1)&&(e1(m)<e(m))
                            e(m)=e1(m);
                            v(m)=i2;
                        elseif (type==2)&&(e2(m)<e(m))
                            e(m)=e2(m);
                            v(m)=i2;

                        end

                    end
                end

                if temp==0
                    break
                end

                [k2,nn2]=find(tour==v(m));
                ej(m,1)=tour(k2,nn2-1);
                ej(m,2)=v(m);
                ej(m,3)=tour(k2,nn2+1);
                ovc1(k2)=ovc1(k2)-d(v(m))+d(v(m-1));
                L1(k2)=L1(k2)-td(ej(m,1),ej(m,2))-
td(ej(m,2),ej(m,3))+td(ej(m,1),ej(m-1,2))+td(ej(m-1,2),ej(m,3));

            end

            if delta<-0.001
                fprintf('  ');
                fprintf('                    ',rr);
                fprintf('                    ',type);
```

```
                fprintf('[illegible]',v(1));
                fprintf('[illegible]',v(m1));
                fprintf('[illegible]',m1);
                fprintf('[illegible]',m);

                X(ej(1,1),ej(1,2),k1)=0;
                X(ej(1,2),ej(1,3),k1)=0;
                ovc(k1)=ovc(k1)-d(v(1));
                disp(k1);
                disp(L(k1));
                L(k1)=L(k1)-td(ej(1,1),ej(1,2))-td(ej(1,2),ej(1,3));
                disp(L(k1));

                for m2=2:m1
                    [k2,~]=find(tour==v(m2));
                    X(ej(m2,1),ej(m2-1,2),k2)=1;
                    X(ej(m2-1,2),ej(m2,3),k2)=1;
                    X(ej(m2,1),ej(m2,2),k2)=0;
                    X(ej(m2,2),ej(m2,3),k2)=0;
                    ovc(k2)=ovc(k2)-d(v(m2))+d(v(m2-1));
                     disp(k2);
                    disp(L(k2));
                    L(k2)=L(k2)+td(ej(m2,1),ej(m2-1,2))+td(ej(m2-
1,2),ej(m2,3))-td(ej(m2,1),ej(m2,2))-td(ej(m2,2),ej(m2,3));
                    disp(L(k2));
                end

                if type==1
                    X(ej(1,1),ej(m1,2),k1)=1;
                    X(ej(m1,2),ej(1,3),k1)=1;
                    ovc(k1)=ovc(k1)+d(v(m1));
                     disp(k1);
                    disp(L(k1));

L(k1)=L(k1)+td(ej(1,1),ej(m1,2))+td(ej(m1,2),ej(1,3));

                    disp(L(k1));
                else
                    X(p,q,k3)=0;
                    X(p,ej(m1,2),k3)=1;
                    X(ej(m1,2),q,k3)=1;
                    ovc(k3)=ovc(k3)+d(v(m1));
                     disp(k3);
                     disp(p);
                     disp(q);
                    disp(L(k3));
                    L(k3)=L(k3)-td(p,q)+td(p,ej(m1,2))+td(ej(m1,2),q);
                    disp(L(k3));

                    if ej(1,1)~=ej(1,3)
                        X(ej(1,1),ej(1,3),k1)=1;
                         disp(k1);
                        disp(L(k1));
                        L(k1)=L(k1)+td(ej(1,1),ej(1,3));
                        disp(L(k1));
```

```
    end
end

if max(ovc)>vc
    error('Vehicle capacity is exceeded');
end

if max(L)>tL
    error('Max tour length is exceeded');
end

while ovc(k)==0
    k=k-1;
end

k4=1;
while k4<=k
    if ovc(k4)==0
        if k4<k
            k6=k4;
            while k6<k
                ovc(k6)=ovc(k6+1);
                X(:,:,k6)=X(:,:,k6+1);
                L(k6)=L(k6+1);
                ovc(k6+1)=0;
                X(:,:,k6+1)=0;
                L(k6+1)=0;
                k6=k6+1;
            end

        end

        k=k-1;
    end
    k4=k4+1;
end

tour(:,:)=0;

k5=1;
L2(k5)=0;
for j=1:NoFac
    if Z(j)==1
        while sum(X(j,:,k5))>=1
            j3=1;
            tour(k5,j3)=j;
            j4=j;

            while j3<=sum(sum(X(:,:,k5)))
                [~,j5]=find(X(j4,:,k5)==1);
                j3=j3+1;
                tour(k5,j3)=j5;
                L2(k5)=L2(k5)+td(tour(k5,j3-1),j5);
                j4=j5;
```

```
                                end
                                if abs(L2(k5)-L(k5))>0.001
                                    error('Wrong tour length');
                                end
                                if k5==k
                                    break
                                else
                                    k5=k5+1;
                                    L2(k5)=0;
                                end
                            end
                        end
                    end

                else
                    ovc1=ovc;
                    L1=L;
                    tabu(v(1),v(2))=randi([5,20]);
                end

            end
        end
    end

if max(L)>tL
    error('Max tour length is exceeded');
end

if max(ovc)>vc
    error('Vehicle capacity is exceeded');
end

RE{1}=k;
RE{2}=X;
RE{3}=ovc;
RE{4}=tour;
RE{5}=L;
```